XiVO-doc Documentation

Release 14.08

Avencall

January 30, 2015

XiVO is an application suite developed by Avencall Group, based on several free existing components including Asterisk, and our own developments to provide communication services (IPBX, Unified Messaging, ...) to businesses.

XiVO is free software. Most of its distinctive components, and XiVO as a whole, are distributed under the *GPLv3 license*.

You may also check XiVO blog and XiVO wiki for more information. XiVO documentation is also available as an EPUB file or as a PDF file

CHAPTER 1

Table of Contents

1.1 Introduction

XiVO is a PABX application developed by the Avencall Group based on several free existent components including Asterisk and our own developments. XiVO provides a solution for enterprises who whish to replace or add a telephony service (PABX).

XiVO is free software. Most of its distinctive components, and XiVO as a whole, are distributed under the GPLv3 license. We are also working on a fully Open XiVO Hardware.

For more information, see XiVO and Open Source

1.1.1 XiVO History

XiVO was created in 2005 by Proformatique.

XiVO 1.2 was released on February 3, 2012.

1.2 Installation

1.2.1 Installing the System

Please refer to the section *Troubleshooting* if ever you have errors during the installation.

There are three official ways to install XiVO:

- using the official ISO image
- using a minimal Debian installation and the XiVO installation script
- using a PXE environment (not detailed here)

XiVO can be installed on both virtual (QEMU/KVM, VirtualBox, ...) and physical machines. That said, since Asterisk is sensible to timing issues, you might get better results by installing XiVO on real hardware.

Installing from the ISO image

The ISO image for XiVO 14.08 can be found at http://mirror.xivo.fr/iso/xivo-current. Download the iso, boot from it and follow the instructions on the installation prompt. We suggest that you *choose english as locale when prompted*. At the end of the installation, you can continue by running the *configuration wizard.*

Installing from a minimal Debian installation

XiVO can be installed directly over a **32-bit** Debian Wheezy. When doing so, you are strongly advised to start with a clean and minimal installation of Debian Wheeze. The latest installation image for Debian Wheezy can be found at http://www.debian.org/distrib/.

Once you have your Debian Wheezy properly installed, log into it and download the XiVO installation script:

```
wget http://mirror.xivo.fr/fai/xivo-migration/xivo_install_current.sh
```

And run it:

```
bash xivo_install_current.sh
```

Note: For testing purposes, you can alternatively install the release candidate or developement version of XiVO. Beware that there is no guarantee that these versions will work nor upgrade correctly.

To install the release candidate version:

```
bash xivo_install_current.sh -r
```

To install the developement version:

```
bash xivo_install_current.sh -d
```

Installing from a PXE

You can visit the XiVO blog for more details on how to install from a PXE.

1.2.2 Running the Wizard

After the system installation, you must go through the wizard before being able to use your XiVO. Browse to your server's IP address to start the configuration wizard (For example: http://192.168.1.10)

Language

You first have to select the language you want to use for the wizard.

Figure 1.1: Select the language

License

You then have to accept the *GPLv3 License* under which XiVO is distributed.

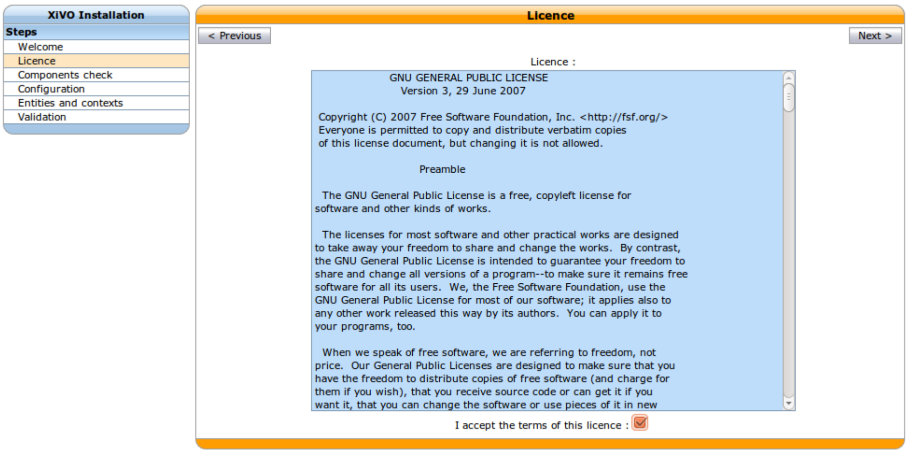


Figure 1.2: Accept the license

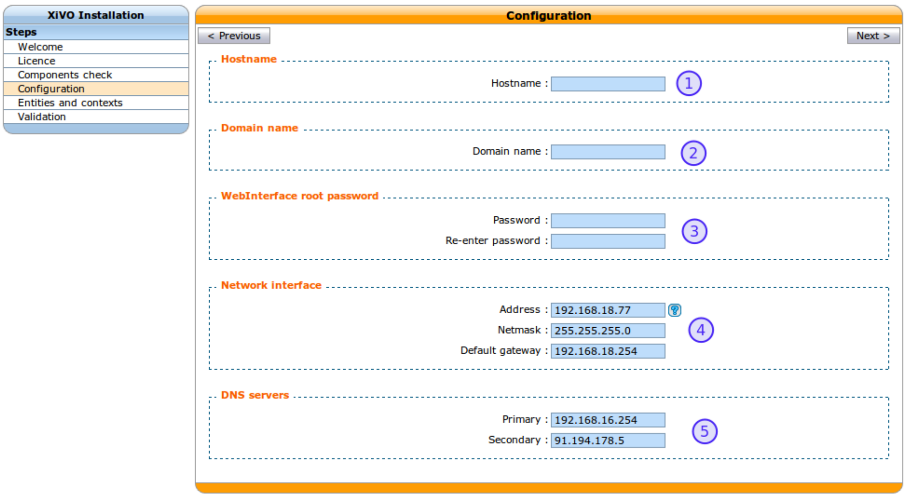


Figure 1.3: Basic configuration

Configuration

1. Enter the hostname (Allowed characters are : `A-Z a-z 0-9 -`)
2. Enter the domain name (Allowed characters are : `A-Z a-z 0-9 - .`)
3. Enter the password for the `root` user of the web interface,
4. Configure the IP address and gateway used by your XiVO (by default it pre-fills the fields with the current IP and gateway of the network interface on which you are connected).

   Note: The network configuration will be applied at the end of the wizard

5. Finally, modify the DNS server information if needed.

Entities and Contexts

Contexts are used for managing various phone numbers that are used by your system.

- The Interal calls context is used for managing phone numbers for devices that are connected to your system.
- The Incalls context will intercept all incoming calls from the exterior
- The Outcalls context is used for managing ougoing calls to the exterior

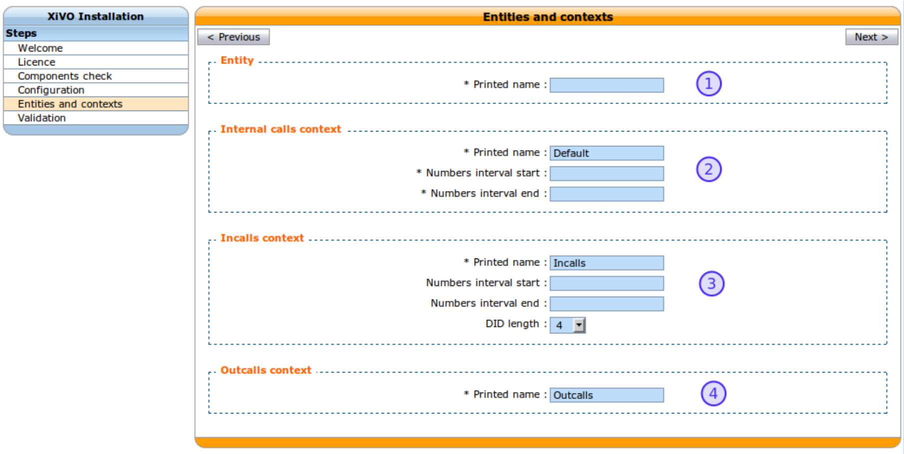


Figure 1.4: Entities and Contexts

1. Enter the entity name (e.g. your organization name) (Allowed characters are : `A-Z a-z 0-9 - .`)
2. Enter the number interval for you internal context. The interval will define the users's phone numbers for your system (you can change it afterwards)
3. Enter the DID range and DID length for your system.
4. You may change the name of your outgoing calls context.

Validation

Finally, you can validate your configuration by clicking on the `Validate` button. Note that if you want to change one of the settings you can go backwards in the wizard by clicking on the `Previous` button.

Warning: This is the last time the `root` password will be displayed. Take care to note it.

Congratulations, you now have a fully functional XiVO server.

You can subscribe to the xivo-announce list to always stay informed on the latest upgrades for XiVO.

1.2.3 Post installation

Display called name on internal calls

When you call internally another phone of the system you would like your phone to display the name of the called person (instead of the dialed number only). To achieve this you must change the following SIP options:

- *Services* → *IPBX* → *General settings* → *SIP Protocol* → *Default*:
  - Trust the Remote-Party-ID: yes,
  - Send the Remote-Party-ID: select `PAI`

Incoming caller number display

The display of caller number on incoming calls depends on what is sent by your operator. You can modify it via the file `/etc/xivo/asterisk/xivo_in_callerid.conf`.

Note: this is this modified caller id number which will be used in the reverse directory lookup

Examples:

- if you use a prefix to dial outgoing numbers (like a 0) you should add a 0 to all the `add =` sections,
- you may want to display incoming numbers in E.164 format. For example, you can change the `[national1]` section to:

  ```
  callerid = ^0[1-9]\d{8}$
  strip = 1
  add = +33
  ```

To enable the changesyou have to restart xivo-agid:

```
service xivo-agid restart
```

Time and date

- Configure your locale and default time zone in device template => *Configuration* → *Provisioning* → *Template Device* by editing the default template
- Configure the Timezone in => *Services* → *IPBX* → *General settings* → *Advanced* → *Timezone*
- Reconfigure your timezone for the system:

  ```
  dpkg-reconfigure tzdata
  ```

Codecs

You should also select default codecs. It obviously depends on the telco links, the country, the phones, the usage etc. Here is a typical example for Europe (the main goal in this example is to select *only* G.711 **A-Law** instead of both G.711 A-Law and G.711 μ-Law by default):

- SIP : *Services* → *IPBX* → *General settings* → *SIP Protocol* → *Signaling*:
  - Customize codec : activate,
  - Disable codec : All
  - Codec list:

```
G.711 A-Law
G.722
G.729A
H.264
```

- IAX2 : *Services* → *IPBX* → *General settings* → *IAX Protocol* → *Default*:
  - Customize : activate,
  - Disallowed codec : All
  - Codec list:

```
G.711 A-Law
G.722
G.729A
H.264
```

1.3 Getting Started

This section will show you how to create a user with a SIP line. This simple use case covers what a lot of people need to start using a phone. You can use these steps for connectinhg a softphone, a Linksys PAP2 or a SIP phone via the web interface.

This tutorial doesn't cover how to automatically provision a supported device. For this, you must refer to the *provisionning section.*

We first need to log into the XiVO web interface. The web interface is where you can administer the whole system.

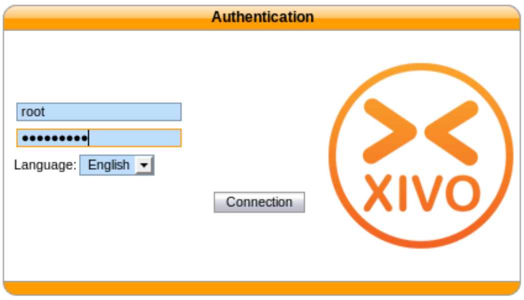


Figure 1.5: Logging into the XiVO

When logged in, you will see a page with all the status information about your system. This page helps you monitor the health of your system and gives you information about your network. Please note the IP address of your server, you will need this information later on when you will configure your device (e.g. phone)

To configure a device for a user, start by navigating to the IPBX menu. Hover over the *Services* tab, a dropdown menu will appear. Click on *IPBX*.

Select the *Users* setting in the left menu.

From here, press on the "plus" sign. A pop up will appear where you can click on *Add*.

We now have the form that will allow us to create a new user. The three most important fields are 'First name', 'Last name' and 'Language'. Fill in the fields and click on *Save* at the bottom. For our example, we will create a used called 'Alice Wonderland'.

Afterwards, click on the "Lines" tab.

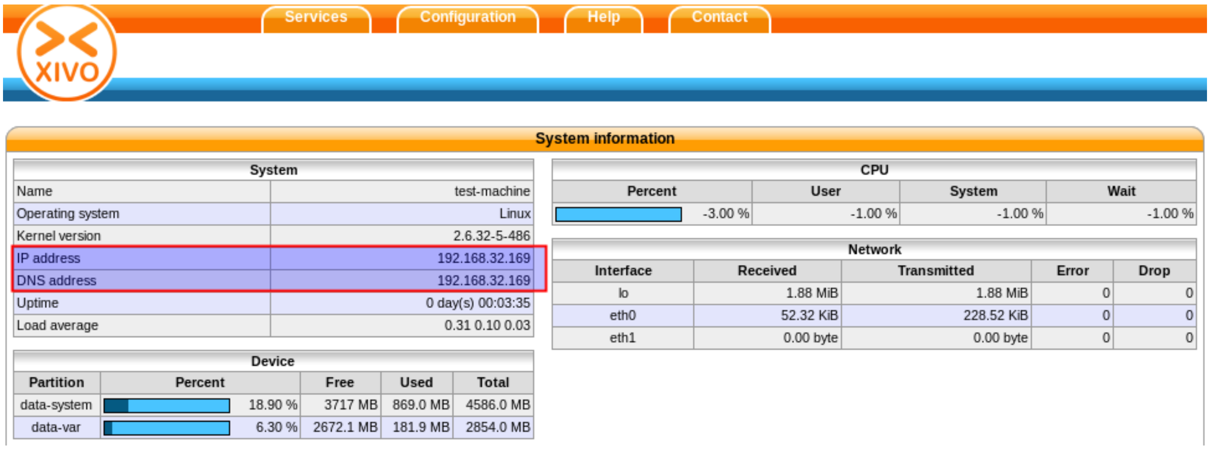


Figure 1.6: System informations

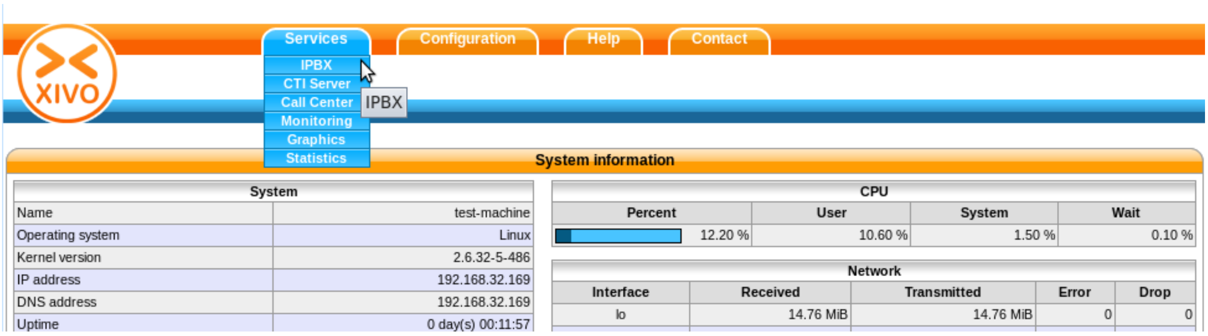


Figure 1.7: Menu IPBX

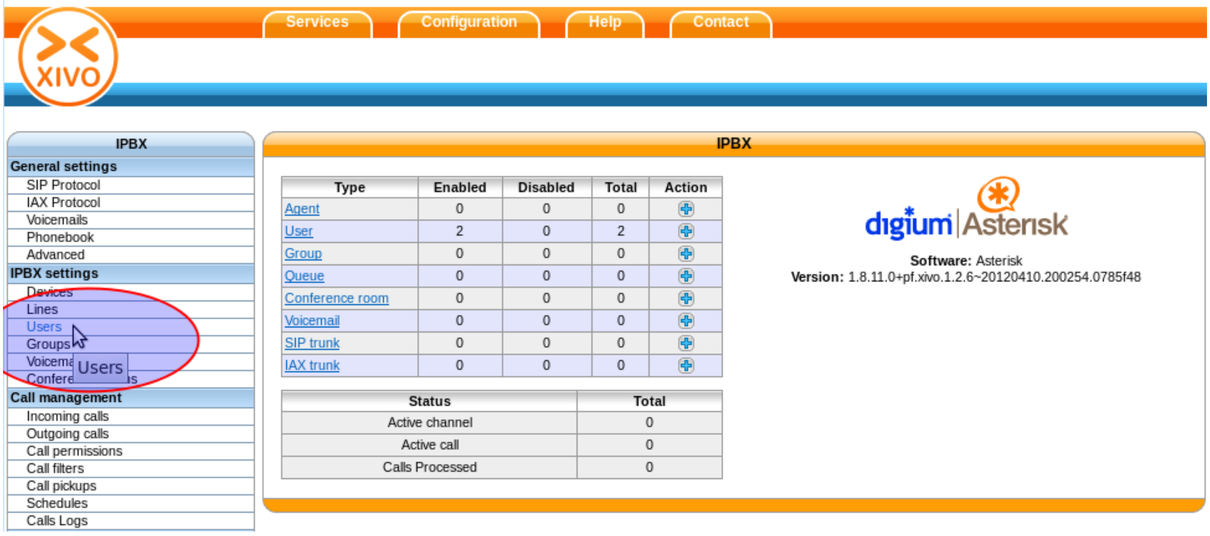


Figure 1.8: Users settings

Figure 1.9: Adding a new line

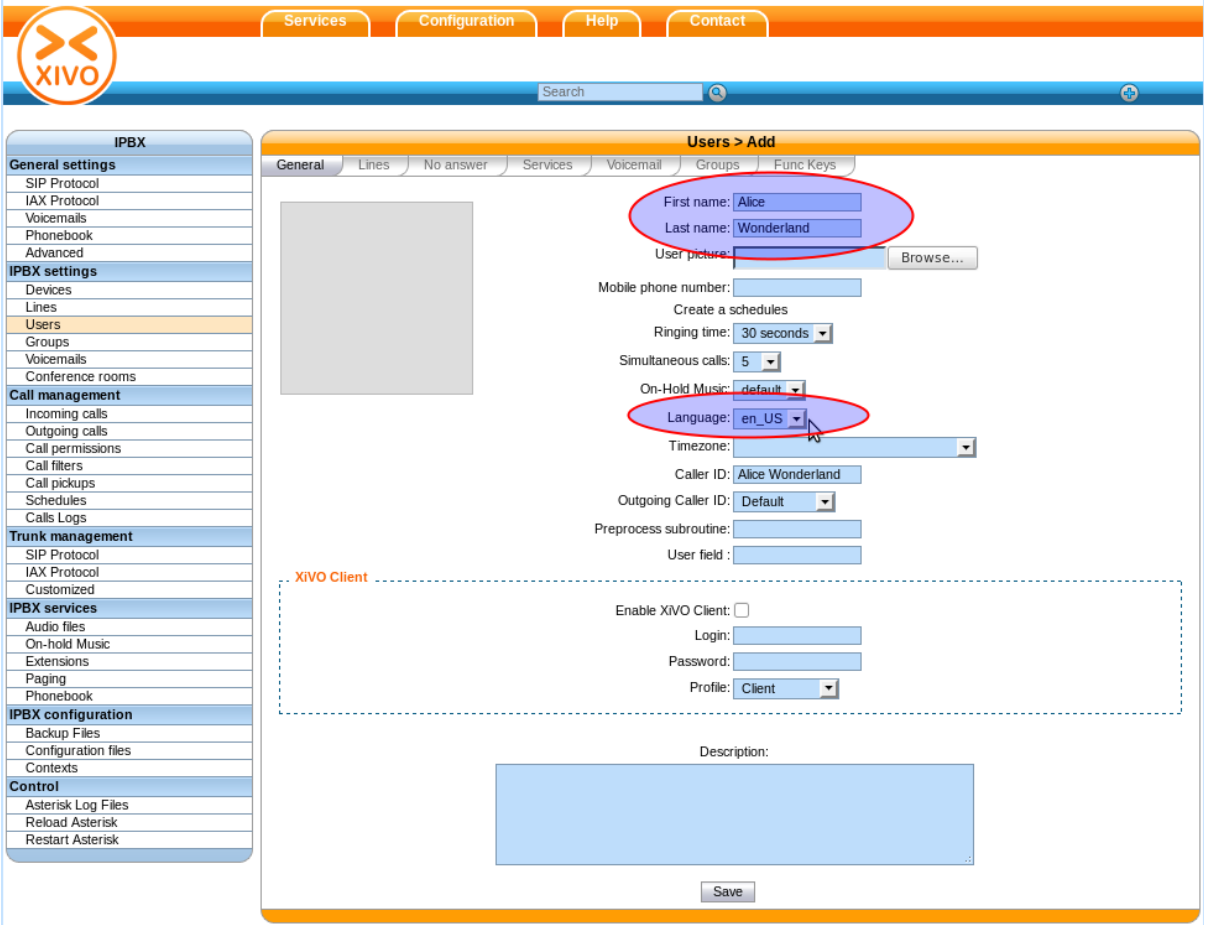


Figure 1.10: User information

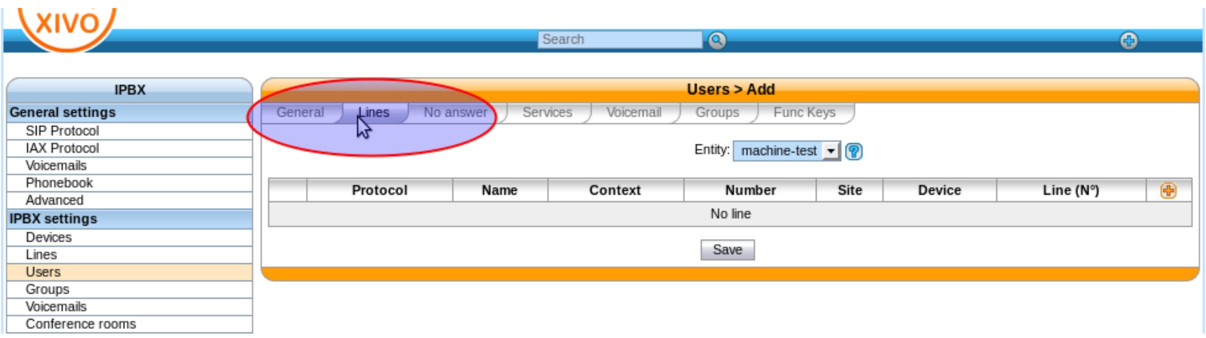


Figure 1.11: Lines menu

Enter a number for your phone. If you click inside the field, you will see the range of numbers you can use. For our example, we will use '1000'.

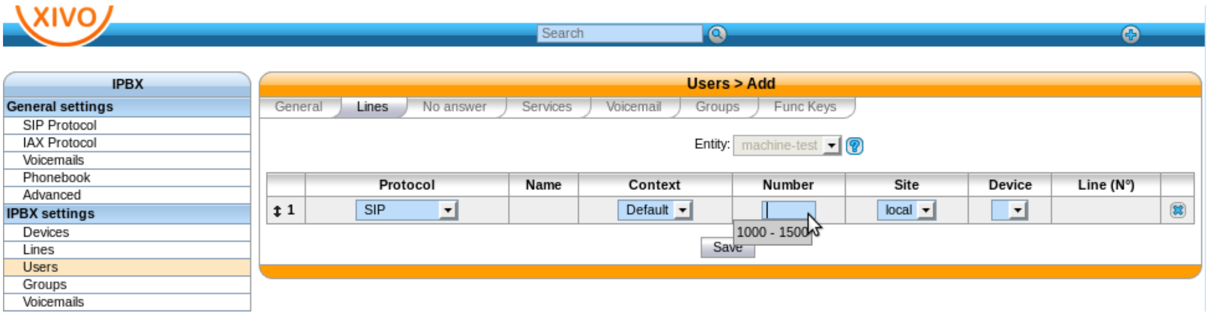


Figure 1.12: Line information

By default, the selected protocol is SIP, which is what we want for now. Click on Save to create the line.

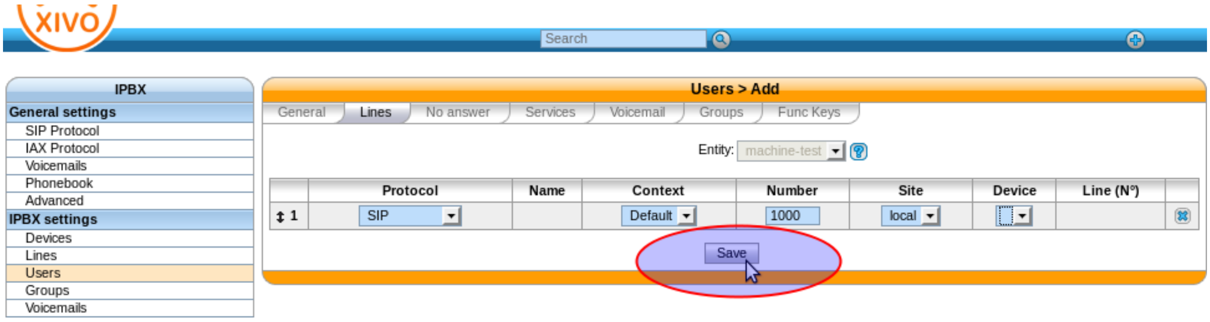


Figure 1.13: Save

Yahoo ! we now have a user named 'Alice Wonderland' with the phone number '1000' !

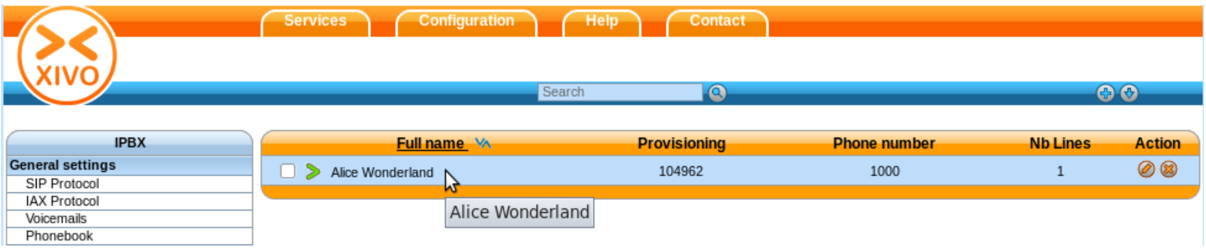


Figure 1.14: User added information

Now we need to go get the SIP username and password to configure our phone. Go back to the IPBX menu on the left, and click on 'Lines'.

You will see a line associated with the user we just created. Click on the pencil icon to edit the line.

We can now see the username and password for the SIP line. you can configure your softphone, your linksys PAP2 or your SIP device by using the IP for your server, the username and the password.

1.4 Upgrading

Upgrading a XiVO is done by executing commands through a terminal on the server. You can connect to the server either through SSH or with a physical console.

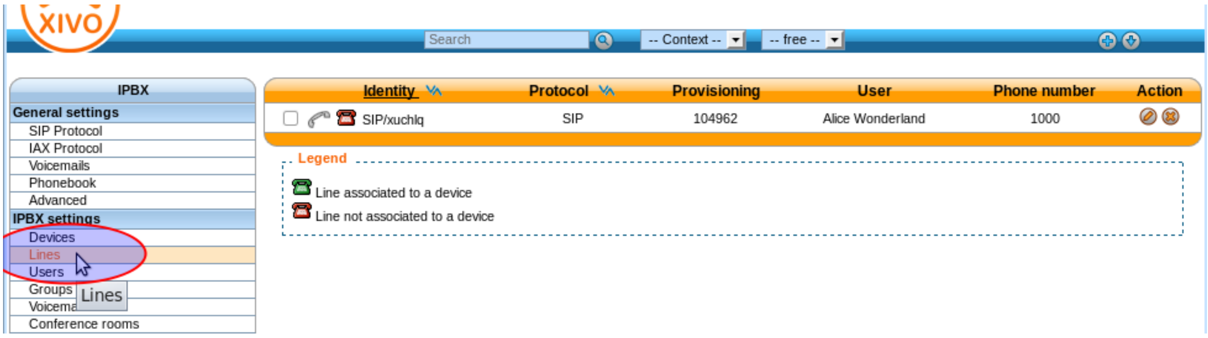


Figure 1.15: Lines information

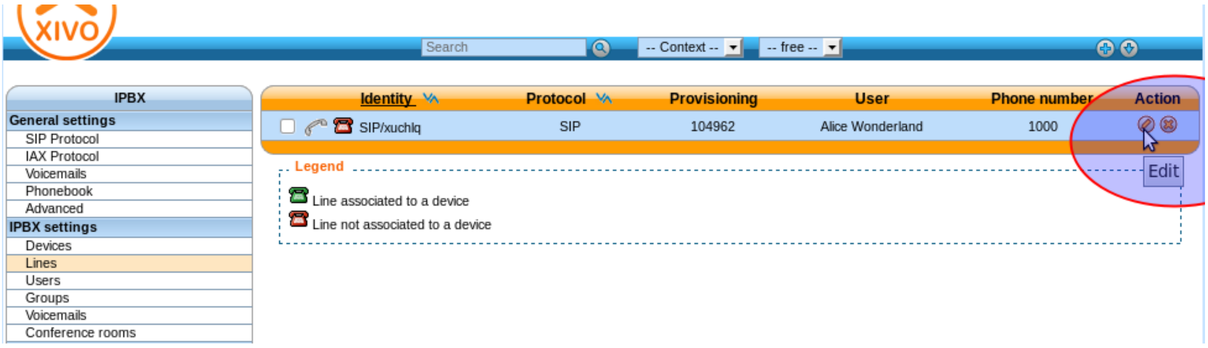


Figure 1.16: Edit line

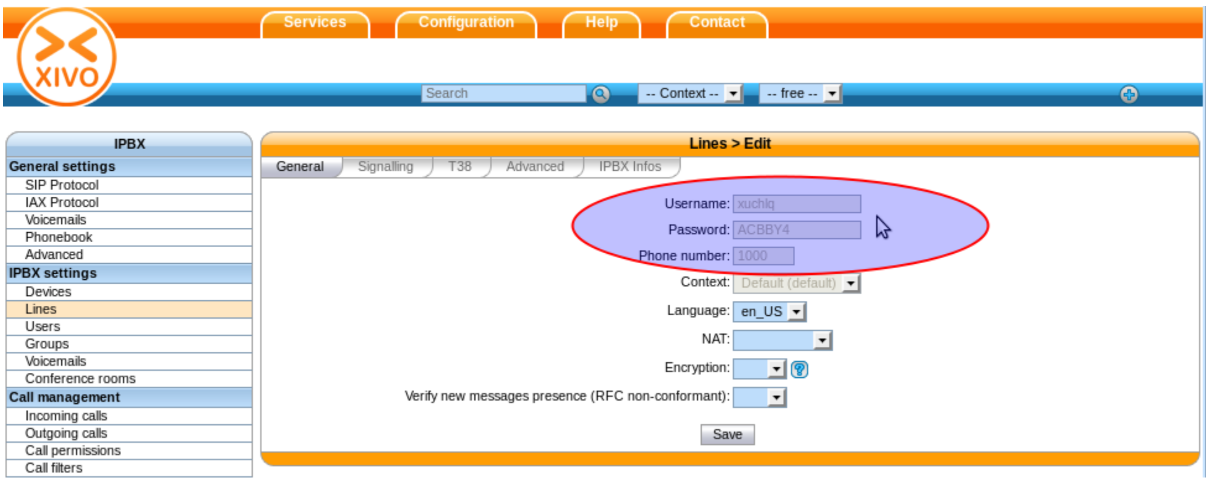


Figure 1.17: General line information

To upgrade your XiVO to the latest version, you **must** use the *xivo-upgrade* script. You can start an upgrade with the command:

```
xivo-upgrade
```

Note:

- You can't use xivo-upgrade if you have not run the wizard yet
- Upgrading to XiVO 1.2 from a previous version (i.e. XiVO 1.1) is not supported right now.
- When upgrading XiVO, you **must** also upgrade **all** associated XiVO Clients. There is currently no retro-compatibility on older XiVO Client versions.

This script will update XiVO and restart all daemons.

There are 2 options you can pass to xivo-upgrade:

- `-d` to only download packages without installing them. **This will still upgrade xivo-upgrade and xivo-service packages**.
- `-f` to force upgrade, without asking for user confirmation

Warning: If xivo-upgrade fails or aborts in mid-process, the system might end up in a faulty condition. If in doubt, run the following command to check the current state of xivo's firewall rules:

```
iptables -nvL
```

If, among others, it displays something like the following line (notice the DROP and 5060)

```
0     0 DROP       udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:5060
```

Then your XiVO will not be able to register any SIP phones. In this case, you must delete the DROP rules with the following command:

```
iptables -D INPUT -p udp --dport 5060 -j DROP
```

Repeat this command until no more unwanted rules are left.

1.4.1 Typical Upgrade Process

- Read all roadmaps starting from your current version to the current prod version.
- Read all existing Upgrade Notes (see below) starting from your current version to the current prod version.
- If in a specific configuration, follow the specific procedure described below (example : cluster).
- To download the packages beforehand, run `xivo-upgrade -d` (will upgrade xivo-upgrade, xivo-service and download all packages necessary, prior to stopping services for upgrade, making the upgrade faster).
- When ready (services will be stopped), run `xivo-upgrade` which will actually start the migration.
- When finished, check that the services are correctly running :
- with `xivo-service status` command,
- and with actual checks like SIP registration, ISDN links status, internal/incoming/outgoing calls, XiVO Client connections etc.

1.4.2 Specific procedure: XiVO 14.01, 14.02, 14.03, 14.04 installed from the ISO file

In those versions, xivo-upgrade keeps XiVO on the same version. You must do the following, before the normal upgrade:

```
echo "deb http://mirror.xivo.fr/debian/ xivo-five main" > /etc/apt/sources.list.d/xivo-upgrade.li
&& apt-get update \
&& apt-get install xivo-fai \
&& rm /etc/apt/sources.list.d/xivo-upgrade.list \
&& apt-get update
```

1.4.3 Specific procedure: XiVO 13.03 and before

When upgrading from XiVO 13.03 or earlier, you must do the following, before the normal upgrade:

```
wget http://mirror.xivo.fr/xivo_current.key -O - | apt-key add -
```

1.4.4 Specific procedure: XiVO 12.13 and before

When upgrading from XiVO 12.13 or earlier, you must do the following, before the normal upgrade:

```
apt-get update
apt-get install debian-archive-keyring
```

1.4.5 Specific procedure: XiVO 1.2.1 and before

Upgrading from 1.2.0 or 1.2.1 requires a special procedure before executing `xivo-upgrade`:

```
apt-get update
apt-get install xivo-upgrade
/usr/bin/xivo-upgrade
```

1.4.6 Specific Procedure : Upgrading a Cluster

Here are the steps for upgrading a cluster:

1. On the master : deactivate the database replication by commenting the cron in `/etc/cron.d/xivo-ha-master`
2. On the slave, deactivate the xivo-check-master-status script cronjob by commenting the line in `/etc/cron.d/xivo-ha-slave`
3. On the slave, start the upgrade:

   ```
   xivo-slave:~$ xivo-upgrade
   ```

4. When the slave has finished, start the upgrade on the master:

   ```
   xivo-master:~$ xivo-upgrade
   ```

5. When done, launch the database replication manually:

   ```
   xivo-master:~$ xivo-master-slave-db-replication <slave ip>
   ```

6. Reactivate the cronjobs (see steps 1 and 2)

1.4.7 Upgrading to/from an archive version

Upgrade involving archive version of XiVO

Archive package names

Archive packages are named as follow:

XiVO version	Archive package name
1.2 to 1.2.12	pf-fai-xivo-1.2-skaro-1.2.1
12.14 to 13.24	xivo-fai-skaro-13.04
from 13.25	xivo-fai-13.25

Upgrade to current version from an archive

```
apt-get update
apt-get install xivo-fai/squeeze-xivo-skaro-$(cat /usr/share/xivo/XIVO-VERSION) xivo-fai-skaro
apt-get update
xivo-upgrade
```

As a result, xivo-upgrade will always upgrade XiVO to the latest stable version.

From the current version, use an archive version

Downgrades are not supported: you can only upgrade to a greater version.

Current version between 1.2 and 13.24:

```
apt-get update
apt-get install xivo-fai-skaro-13.23
apt-get purge xivo-fai-skaro
apt-get update
```

Current version after 13.25:

```
apt-get update
apt-get install xivo-fai-13.25
apt-get purge xivo-fai
apt-get update
```

As a result, xivo-upgrade will not upgrade XiVO to a greater version than the archive you chose.

Upgrade from an archive version to another archive version

Downgrades are not supported: you can only upgrade to a greater version.

Source and destination archive version between 1.2 to 13.24:

```
apt-get update
apt-get install xivo-fai-skaro-13.24
apt-get purge xivo-fai-skaro-13.02
apt-get update
xivo-upgrade
```

Source or destination archive version after 13.25:

```
apt-get update
echo "deb http://mirror.xivo.fr/archive xivo-13.25 main" > /etc/apt/sources.list.d/xivo-13.25.lis
apt-get update
apt-get install xivo-fai-13.25
```

```
rm /etc/apt/sources.list.d/xivo-13.25.list
apt-get purge xivo-fai-skaro-13.02
apt-get update
xivo-upgrade
```

1.4.8 Upgrade Notes

14.08

- Consult the 14.08 Roadmap
- The `xivo` database has been merged into the `asterisk` database. The database schema has also been altered in a way that it might make the upgrade longer than usual.

Please consult the following detailed updated notes for more information:

Databases Merge Upgrade Notes

The `xivo` database has been merged into the `asterisk` database in XiVO 14.08. This has an impact on:

- The *restore* procedure. There's only one database to restore now. Also, the procedure to restore the data while keeping the system configuration has been updated.
- The data that is replicated between the master and the slave in a *high availability* cluster.

Previously, all the configuration that was under the "Configuration" menu of the web interface was not replicated between the master and slave. This is now replicated, except for:

- HA settings
- All the network configuration (i.e. everything under the *Configuration* → *Network* section)
- All the support configuration (i.e. everything under the *Configuration* → *Support* section)

The call center statistics have also been excluded from the replication.

The way the replication is done has also been updated, which makes it faster.

Optional Upgrade Procedure When upgrading to XiVO 14.08, the database schema will be altered.

This will result in a longer upgrade time if you have a lots of rows in the queue_log table.

You can see the number of rows in your queue_log table with:

```
sudo -u postgres psql -c "SELECT count(*) FROM queue_log" asterisk
```

On ordinary hardware, you can expect that it will take ~10 minutes for every 2.5 million of rows. So if you have 5 million of rows in your queue_log table, you can expect that the upgrade will take an extra 20 minutes.

It is possible to reduce the amount of additional time the upgrade will take by either removing rows from the table or altering the table before the upgrade.

Both these commands can be run while the XiVO services are up.

For example, if you want to remove all the rows before march 2014, you can use:

```
sudo -u postgres psql -c "DELETE FROM queue_log WHERE \"time\" < '2014-03-01'" asterisk
```

If you want to alter the table before the upgrade, you can use:

```
sudo -u postgres psql -c "ALTER TABLE queue_log ADD COLUMN id SERIAL PRIMARY KEY; GRANT ALL ON SE
```

Note: It is recommended to execute this command when there's no activity on the system.

More Technical Information The way the database is initially provisioned and the way it is altered during an upgrade has also been changed.

In XiVO 14.07 and earlier, the database was provisioned by executing the `/usr/share/xivo-manage-db/datastorage/asterisk.sql` SQL script. Starting with XiVO 14.08, the `xivo-init-db` is responsible for provisioning the database. This script should not be used by an administrator in normal circumstance.

Starting with XiVO 14.08, database migration are done with the help of alembic instead of the asterisk-XXX.sql and xivo-XXX.sql scripts. The alembic migration scripts can be found inside the `/usr/share/xivo-manage-db` directory.

Otherwise, the `xivo-check-db` and `xivo-update-db` commands have been updated to work with both the old and the new systems and are still the official way to check the database state and update the database respectively.

14.07

- Consult the 14.07 Roadmap
- Configuration for phones used for the switchboard has changed.

Please consult the following detailed updated notes for more information:

Switchboard Phone Configuration Upgrade Notes

The xivo-aastra-switchboard and xivo-snom-switchboard plugins have been removed and their functionalities are now provided by the generic xivo-aastra and xivo-snom plugins respectively.

The upgrade is not done automatically, so please follow the Upgrade Procedure section below.

Although you are strongly advised to upgrade your switchboard phone configuration, backwards compatibility with the old system will be maintained.

Note that if you need to install a switchboard for a previous version of XiVO, the old xivo-aastra-switchboard and xivo-snom-switchboard plugins can be found in *the archive repository*.

Upgrade Procedure This procedure should be executed after the upgrade to 14.07 or later: the options used in this procedure are not available in versions before 14.07.

The following upgrade procedure suppose that you are using an Aastra phone as your switchboard phone. The same upgrade procedure apply for Snom phones, with the only difference being the different plugin name.

1. Update the list of installable plugins.
2. Install the latest xivo-aastra plugin, or upgrade it to the latest version if it is already installed.
3. Install the needed language files and firmware files.
4. For each phone used for the switchboard, *change the plugin and activate the switchboard option*:
   - Select the generic xivo-aastra plugin.
   - Check the "switchboard" checkbox.
   - Synchronize the phone.
5. Once this is completed, you can uninstall the xivo-aastra-switchboard plugin.

An unofficial script that automates this procedure is also available on github:

```
cd /tmp
wget --no-check-certificate https://raw.githubusercontent.com/xivo-pbx/xivo-tools/master/scripts/r
python migrate_switchboard_1407.py
```

14.06

- Consult the 14.06 Roadmap
- The XiVO client now uses Qt 5 instead of Qt 4. There is nothing to be aware of unless you are *building your own version* of it.

14.05

- Consult the 14.05 Roadmap
- The *CTI Protocol* has been updated.
- The specification of the 'answered-rate' queue statistic has changed to exclude calls on a closed queue
- The switchboard can now choose which incoming call to answer
- The package versions do not necessarily contain the current XiVO version, it may contain older versions. Only the package `xivo` is guaranteed to have the current XiVO version.

Please consult the following detailed updated notes for more information:

DAHDI 2.9.0 Upgrade Notes

These notes only apply to Digium TE133 or TE134 cards that are in firmware version 770017 or earlier.

Warning: The system will need to be power cycled after the upgrade. Your cards will not be usable until then.

After the upgrade First, you need to install the latest firmware for your TE133 or TE134 cards:

```
xivo-fetchfw install digium-te133
xivo-fetchfw install digium-te134
```

Then stop all the services and reload the DAHDI modules. Reloading the DAHDI module might take up to 30 seconds:

```
xivo-service stop
service dahdi stop
service dahdi start
```

Following this manipulation, you should see something similar at the end of the `/var/log/messages` file:

```
dahdi: Telephony Interface Unloaded
dahdi: Version: 2.9.0
dahdi: Telephony Interface Registered on major 196
wcte13xp 0000:03:0c.0: Firmware version 6f0017 is running, but we require version 780017.
wcte13xp 0000:03:0c.0: firmware: agent loaded dahdi-fw-te134.bin into memory
wcte13xp 0000:03:0c.0: Found dahdi-fw-te134.bin (version: 780017) Preparing for flash
wcte13xp 0000:03:0c.0: Uploading dahdi-fw-te134.bin. This can take up to 30 seconds.
wcte13xp 0000:03:0c.0: Delaying reset. Firmware load requires a power cycle
wcte13xp 0000:03:0c.0: Running firmware version: 6f0017
wcte13xp 0000:03:0c.0: Loaded firmware version: 780017 (Will load after next power cycle)
wcte13xp 0000:03:0c.0: FALC version: 5
wcte13xp 0000:03:0c.0: Setting up global serial parameters for T1
wcte13xp 0000:03:0c.0: VPM450: firmware dahdi-fw-oct6114-032.bin not available from userspace
wcte13xp 0000:03:0c.0: Found a Wildcard TE132/TE134 (SN: 1TE134F - DF05132600690 - B1 - 20130702)
```

For the firmware update to complete, you **must halt** the machine (a reboot won't be enough) before restarting it.

SCCP Upgrade Notes

Important modification have been made to the internal structure of the SCCP channel driver, xivo-libsccp.

The modifications mostly affect administrators; users are not affected.

Major changes are:

- Improved support for live modifications; no more manual intervention in the asterisk CLI is needed.
- Improved handling of concurrency; crash and deadlock due to concurrency problems should not occur anymore.

CLI The following commands have been removed because they were not needed:

- sccp resync
- sccp set directmedia
- sccp show lines
- sccp update config

The behavior of the following commands have been changed:

- `module reload chan_sccp` reloads the module configuration, without interrupting the telephony service. A device will only be resetted/restarted if needed, and only once the device is idle. Some changes don't even require the device to be resetted.
- `sccp show config` output format has been changed a little.
- `sccp show devices` only show the connected devices instead of all the devices. This might change in the future. To get a list of all the devices, use `sccp show config`.

Configuration File The format of the `sccp.conf` configuration file has been changed. This will only impact you if you are using xivo-libsccp without using XiVO.

The format has been changed because the module is now using the ACO module from asterisk, which expect configuration file to have a specific format.

See sccp.conf.sample for a configuration file example.

Other Each SCCP session/connection now use 3 file descriptors instead of 1 previously. On XiVO, the file descriptor limit for the asterisk process is 8192, which means that the increase in used file descriptors should not be a problem, even on a large installation.

14.04

- Consult the 14.04 Roadmap
- Live reload of the configuration can be enabled and disabled using the REST API
- The generation of call logs for unanswered calls from the XiVO client have been improved.

14.03

- Consult the 14.03 Roadmap
- A migration script adds an index on the linkedid field in the cel table. Tests have shown that this operation can last up to 11.5 minutes on a XiVO Corporate with 18 millions CELs. xivo-upgrade will thus be slightly longer.
- Two new daemons are now operationnal, xivo-amid and xivo-call-logd:

- xivo-amid constantly reads the AMI and sends AMI events to the RabbitMQ bus
- xivo-call-logd generates call-logs in real time based on AMI LINKEDID_END events read on the bus

- An increase in load average is expected with the addition of these two new daemons.
- The cron job calling xivo-call-logs now runs once a day at 4:25 instead of every 5 minutes.

14.02

- Consult the 14.02 Roadmap
- PHP Web services has been removed from documentation
- REST API 1.0 Web services has been removed from documentation
- REST API 1.1 User-Line-Extension service is replaced by User-Line and Line-Extension services

14.01

- Consult the 14.01 Roadmap
- The following paths have been renamed:
  - `/etc/pf-xivo` to `/etc/xivo`
  - `/var/lib/pf-xivo` to `/var/lib/xivo`
  - `/usr/share/pf-xivo` to `/usr/share/xivo`

You must update any dialplan or configuration file using these paths

Archives

Archived Upgrade Notes

2013

13.25

- Consult the 13.25 Roadmap
- Debian has been upgraded from version 6 (Squeeze) to 7 (Wheezy), which is essentially a complete system upgrade.

Please consult the following detailed upgrade notes for more information:

Debian GNU/Linux Wheezy Upgrade Notes

Before the upgrade

- The upgrade will take longer than usual, because the whole Debian system will be upgraded
- The system must be restarted after the upgrade, because the Linux kernel will also be upgraded

LDAPS In case XiVO is using a LDAP server through SSL/TLS (LDAPS), the documentation instructed you to append the certificate to `/etc/ssl/certs/ca-certificates.crt`. However, this is the wrong way to add a new certificate, because it will be erased by the upgrade.

To keep your certificate installed through the upgrade, you must follow the instructions given in the *LDAP documentation*.

After the upgrade

GRUB (Cloned Virtual Machines only) GRUB installations on cloned virtual machines may lead to unbootable systems, if not fixed properly before restarting the system. If xivo-upgrade detects your system is in a broken state, it will display a few commands to repair the GRUB installation.

13.24

- Consult the 13.24 Roadmap
- Default Quality of Service (QoS) settings have been changed for SCCP. The IP packets containing audio media are now marked with the EF DSCP.

13.23

- Consult the 13.23 Roadmap
- The *New call* softkey has been removed from SCCP phones in *connected* state. To start a new call, the user will have to press *Hold* then *New call*. This is the same behavior as a *Call Manager*.
- Some softkeys have been moved on SCCP phones. We tried to keep the keys in the same position at any given time. As an example, the *transfer* key will not become *End call* while transfering a call. Note that this is a work in progress and some models still need some tweaking.

13.22

- Consult the 13.22 Roadmap
- PostgreSQL will be upgraded from 9.0 to 9.1. The upgrade of XiVO will take longer than usual, depending on the size of the database. Usually, the database grows with the number of calls processed by XiVO. The upgrade will be stopped if not enough space is available on the XiVO server.

13.21

- Consult the 13.21 Roadmap
- It is no more possible to delete a device associated to a line using REST API.

13.20

- Consult the 13.20 Roadmap
- xivo-libsccp now supports direct media on wifi phone 7920 and 7921
- xivo-restapi now implements a voicemail list

13.19

- Since XiVO 13.18 was not released, the 13.19 release contains all developments of both 13.18 and 13.19, therefore please consult both Roadmaps :
- Consult the 13.19 Roadmap
- Consult the 13.18 Roadmap
- Call logs are now generated automatically, incrementally and regularly. Call logs generated before 13.19 will be erased one last time.
- The database was highly modified for everything related to devices : table devicefeatures does not exist anymore and now relies on information from xivo-provd.

13.17

- Consult the 13.17 Roadmap
- There is a major change to call logs. They are no longer available as a web report but only as a csv export. See the *call logs documentation*. Furthermore, call logs are now fetched with the new REST API. See *Call Logs*.
- Paging group numbers are now exclusively numeric. All non-numeric paging group numbers are converted to their numeric-only equivalent while upgrading to XiVO 13.17 (*58 becomes 58, for example).

13.16

- Consult the 13.16 Roadmap
- A migration script modifies the user and line related-tables and the way users, lines and extensions are associated. As a consequence of this script, it is not possible any more to associate a user and a line without extensions. Existing associations between users and one or more lines having no extensions will be removed. Users and lines will still exist unassociated.
- The *call logs* page is able to display partial results of big queries, instead of displaying a blank page.
- Two new CEL messages are now enabled : LINKEDID_END and BRIDGE_UPDATE. Those events will only exist in CEL for calls passed after upgrading to XiVO 13.16.
- The new REST API now makes possible to associate multiple user to a given line and/or extension. There are currently some limitations on how those users and lines can be manipulated using the web interface. Please read the *REST API 1.1 documentation* and more precisely the *Associate Line to User* section for more information.

13.15

- There was no production release of XiVO 13.15. All 13.15 developments are included in the official 13.16 release.

13.14

- Consult the 13.14 Roadmap
- The latest Polycom plugin enables the phone lock feature with a default user password of '123'. All Polycom phones used with XiVO also have a default admin password. In order for the phone lock feature to be secure, one should change every phone's admin AND user passwords.
- WebServices for SIP trunks/lines: field `nat`: value `yes` changed to `force_rport,comedia`
- The database has beed updated in order to remove deprecated tables (generalfeatures, extenumbers, extenhash, cost_center).

13.13

- Consult the 13.13 Roadmap

13.12

- Consult the 13.12 Roadmap
- CTI protocol: Modified values of agent `availability`. Read *CTI Protocol changelog*
- Clean-up was made related to the minimization of the XiVO Client. Some visual differences have been observed on Mac OS X that do not affect the XiVO Client in a functional way.

13.11

- Consult the 13.11 Roadmap
- Asterisk has been upgraded from version 11.3.0 to 11.4.0

API changes:

- Dialplan variable XIVO_INTERFACE_0 is now XIVO_INTERFACE
- Dialplan variable XIVO_INTERFACE_NB and XIVO_INTERFACE_COUNT have been removed
- The following fields have been removed from the lines and users web services
  - line_num
  - roles_group
  - rules_order
  - rules_time
  - rules_type

13.10

- Consult the 13.10 Roadmap

API changes:

- CTI protocol: for messages of class `getlist` and function `updateconfig`, the `config` object/dictionary does not have a `rules_order` key anymore.

13.09

- Consult the 13.09 Roadmap
- The *Restart CTI server* link has been moved from *Services* → *CTI Server* → *Control* to *Services* → *IPBX* → *Control*.
- The Agent Status Dashboard has been optimized.
- The Directory xlet can now be used to place call.

13.08

- Consult the 13.08 Roadmap
- asterisk has been upgraded from version 1.8.21.0 to 11.3.0, which is a major asterisk upgrade.
- The switchboard's queue now requires the *xivo_subr_switchboard* preprocess subroutine.
- A fix to bug #4296 introduced functional changes due to the order in which sub-contexts are included. Please refer to ticket for details.

Please consult the following detailed upgrade notes for more information:

Asterisk 1.8 to 11 Upgrade Notes Table of modules that were available in the asterisk 1.8 package but that are not available anymore in the asterisk 11 package:

Name	Description	Loaded in AST1.8	Asterisk Status	Replaced By
app_dahdibarge	Barge in on DAHDI channel application	Yes	Deprecated	app_chanspy
app_readfile	Stores output of file into a variable	Yes	Deprecated	func_env (FILE())
app_saycountpl	Say polish counting words	Yes	Deprecated	say.conf
app_setcallerid	Set CallerID Presentation Application	Yes	Deprecated	func_callerid
cdr_sqlite	SQLite CDR Backend	No	Removed	cdr_sqlite3_custom
chan_gtalk	Gtalk Channel Driver	No	Deprecated	chan_motif
chan_jingle	Jingle Channel Driver	No	Deprecated	chan_motif
chan_vpb	Voicetronix API driver	No	Supported	
format_sln16	Raw Signed Linear 16KHz Audio support	Yes	Removed	format_sln
res_ais	SAForum AIS	No	Removed	res_corosync
res_jabber	AJI - Asterisk Jabber Interface	No	Deprecated	res_xmpp

List of modules that were loaded in asterisk 1.8 but that are not loaded anymore in asterisk 11 (see modules.conf):

- res_calendar.so
- res_calendar_caldav.so
- res_calendar_ews.so
- res_calendar_exchange.so
- res_calendar_icalendar.so
- res_config_sqlite.so
- res_stun_monitor.so

List of debian packages that are not available anymore for asterisk 11:

- asterisk-config
- asterisk-mysql
- asterisk-web-vmail

Note: These packages were not installed by default for asterisk 1.8.

If you are using some custom dialplan or AGIs, it is your responsibility to make sure it still works with asterisk 11. See the External Links for more information.

External Links

- http://svnview.digium.com/svn/asterisk/branches/11/UPGRADE-10.txt
- http://svnview.digium.com/svn/asterisk/branches/11/UPGRADE.txt
- https://wiki.asterisk.org/wiki/display/AST/New+in+10
- https://wiki.asterisk.org/wiki/display/AST/New+in+11

The switchboard's queue preprocess subroutine The switchboard's queue now uses a preprocess subroutine named *xivo_subr_switchboard*. This preprocess subroutine will be associated with all queues named *__switchboard* that have no preprocess subroutine defined before the upgrade.

If your switchboard queue is named anything other than *__switchboard* you should add the preprocess subroutine manually.

If your switchboard queue already has a preprocess subroutine, you should add a Gosub(xivo_subr_switchboard) to you preprocess subroutine.

Warning: This change is only applied to the switchboard distribution queue, not the queue for calls on hold.

13.07

- Consult the 13.07 Roadmap
- Agent Status Dashboard has more features and less limitations. See related *agent status dashboard documentation*
- XiVO call centers have no more notion of 'disabled agents'. All previously disabled agents in web interface will become active agents after upgrading.
- asterisk has been upgraded from version 1.8.20.1 to 1.8.21.0. Please note that in XiVO 13.08, asterisk will be upgraded to version 11.
- DAHDI has been upgraded from version 2.6.1 to 2.6.2.
- libpri has been upgraded from version 1.4.13 to 1.4.14.
- PostgreSQL upgraded from version 9.0.4 to 9.0.13

13.06

- Consult the 13.06 Roadmap
- The new Agent Status Dashboard has a few known limitations. See related *dashboard xlet known issues section*
- Status Since counter in xlet list of agents has changed behavior to better reflect states of agents in queues as seen by asterisk. See Ticket #4254 for more details.

13.05

- Consult the 13.05 Roadmap
- The bug #4228 concerning BS filter only applies to 13.04 servers installed from scratch. Please upgrade to 13.05.
- The order of softkeys on SCCP phones has changed, e.g. the *Bis* button is now at the left.

13.04

- Consult the 13.04 Roadmap
- Upgrade procedure for HA Cluster has changed. Refer to *Specific Procedure : Upgrading a Cluster*.
- Configuration of switchboards has changed. Since the directory xlet can now display any column from the lookup source, a display filter has to be configured and assigned to the __switchboard_directory context. Refer to *Directory xlet documenttion*.
- There is no more context field directly associated with a call filter. Boss and secretary users associated with a call filter must necessarily be in the same context.

2012

12.24

- Consult the 12.24 Roadmap
- XiVO 12.24 has some limitations mainly affecting the contact center features due to the rewriting of the code handling agents.

Please consult the following detailed upgrade notes for more information:

Contact Center XiVO 12.24 In order to fix problems related to Asterisk freezing through the chan_agent module, XiVO 12.24 implements a new way of managing agents.

Warning The contact center XiVO 12.24 does not implement all the features available in 12.22. Therefore, you must not upgrade your XiVO if you depend on these features. These features will be reimplemented in the future starting with version 13.01.

Missing Features

- Skill-based routing
- Penalities
- Call listening

Live reload via the web interface Agents must be logged out for the following operations:

- Adding or removing agents from the queues
- When changing the name of a queue (only the name, not the displayed name)

You can logoff all the agents with the following command:

```
xivo-agentctl -c "logoff all"
```

Preprocess subroutines Subroutines on users are currently no longer executed when an agent receives a call from the queue

High availability (HA) HA for the contact center is not supported for the moment. When switching from a master to a slave, you must relog all your agents.

SCCP Devices The “Available” / “In use” statuses for agents that are logged in do not work for the moment.

Changes in behavior

“In use” indicator in the XiVO client In XiVO 12.22, an agent is seen as “In use” when:

- The agent’s phone is ringing or has answered a call coming only from a queue

In XiVO 12.24:

- The agent’s phone is “In use” no matter where the call comes from

“Available” indicator in the XiVO client In XiVO 12.22, an agent is seen as “Available” when:

- The agent is not in pause/wrapup and his phone isn’t ringing/in conversation for a call coming from a queue

In XiVO 12.24:

- The agent is not in pause/wrapup and his phone is in the “idle” state

“Agent linked” / “Agent unlinked” Events The “Agent linked” event no longer exists in XiVO 12.24. xivo-upgrade will automatically migrate “Agent linked” / “Agent unlinked” sheets to the “linked” / “unlinked” event.

“Static” Agent VS “Dynamic” agent in the XiVO client There is no longer a difference between a “static” or “dynamic” membership. All agent memberships are now considered “static”. Membership changes between the web interface and the XiVO client are now synchronized.

Updates Please note that when upgrading, the following actions will take place automatically:

- All agents will be logged off before migrating
- All agents with a "dynamic" membership will be removed from their queues

Useful links *CTI server is frozen and won't come back online*

Another change is in effect beginning with XiVO 12.24: the field `profileclient` in the CSV user import sees its values change.

Old value	New value
client	Client
agent	Agent
switchboard	Switchboard
agentsup	Supervisor
oper	*removed*
clock	*removed*

1.5 CTI Client

This section describes the CTI Client and its various xlets

1.5.1 Getting the XiVO Client

Binaries of the XiVO Client are available on our mirror.

> **Warning:** The installed version of the XiVO Client must match the XiVO server's version installation. With our current architecture, there is no way to guarantee that the XiVO server will be retro-compatible with older versions of the XiVO Client. Non-matching XiVO server and XiVO Clients versions will bring inconsistencies.

Choose the version you want and in the right directory, get :

- the `.exe` file for Windows
- the `.deb` file for Ubuntu or Debian (i386 or amd64, depending on your computer)
- the `.dmg` file for Mac OS

For Windows, double-click on the file and follow the instructions.

For Ubuntu/Debian, double-click on the file or execute the following command:

```
$ gdebi xivoclient-*.deb
```

For Mac OS, double-click on the file and drag-and-drop the inner file on the Application entry of the Finder.

The XiVO Client should then be available in the applications menu of each platform.

1.5.2 Connection to the server

To connect to the server using the XiVO client you need a user name, a password and the server's address. Optionally, it is possible to login an agent while connecting to the server. An option is available in the configuration, account to show agent login info.

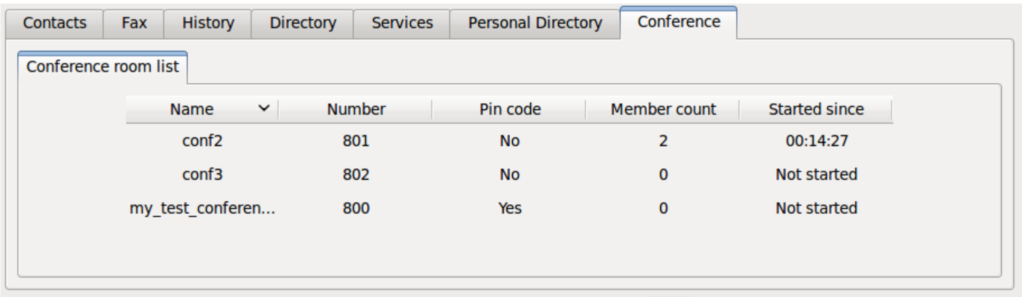
1.5.3 Xlets

Xlets are features of the CTI Client. It is the contraction of XiVO applets.

Conference Xlet

Overview

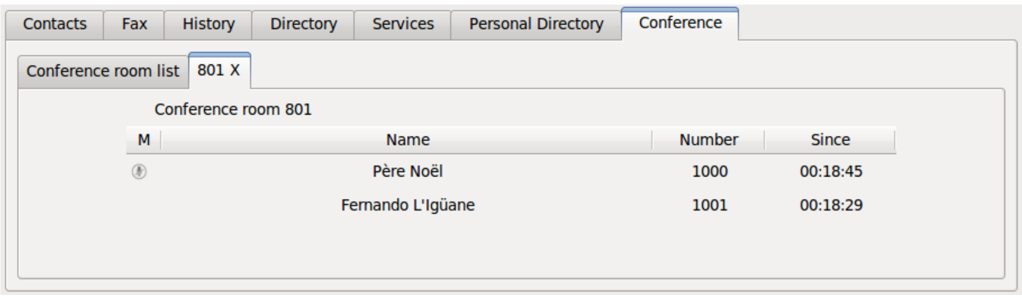
The conference xlet allow the user to join conferences and view conference room statuses.



Usage

The *Conference room list* tab show all available conference rooms configured on the XiVO. The user can right click on a conference to join the conference. When a user joins a conference, his phone will ring and the conference will be joined when the user answers the phone.

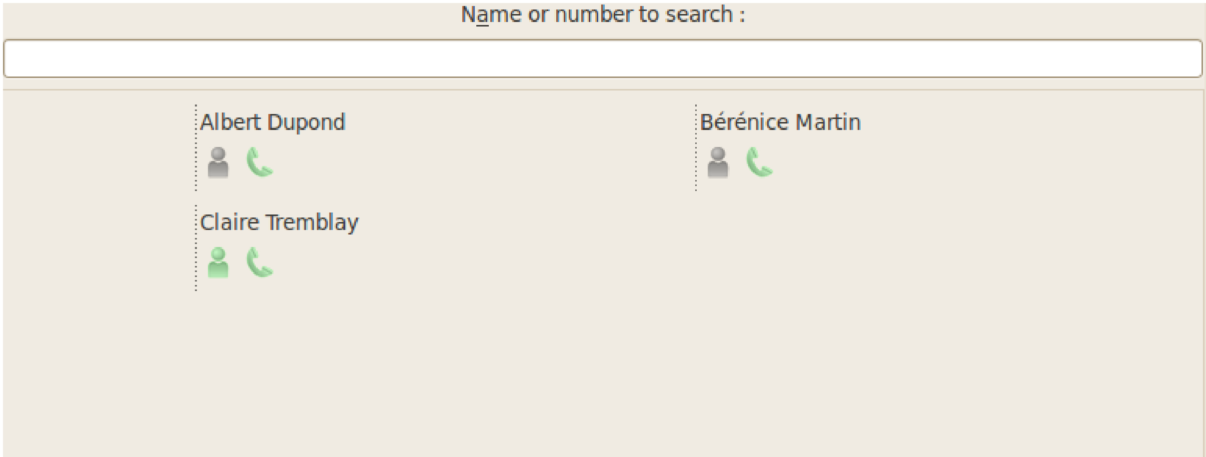
When left clicking on a conference room a new tab is opened for the selected conference room. The new tab contains information about the members of the conference. The name and number of the member will be displayed when available. Users can also mute and unmute themselves using the microphone icon on the left.



Contact Xlet

Overview

The Contacts XLet lists the people of your company, giving you access to their phone and XiVO Client status.



Usage

The "Search" text input allows you to filter the list of people according to their name or phone number. An empty filter displays all contacts found.

If the filter matches some contacts, you can see contact objects. Here's what a contact object looks like :

You can see three informations :

- The person's full name
- A person icon: it displays the status of its XiVO Client. Usually:
  - green means connected
  - gray means disconnected
- A phone icon: it displays the status of the phone of the user, if it has a configured phone. Usually:
  - green means that the phone is activated and hanged up
  - red means that the phone is activated and in communication
  - white means that the phone is not registered, i.e. not functional.

The colors and label of these two statuses may be configured within the XiVO Web interface.

You can interact in several ways with a person object :

- Holding your mouse cursor on the person or phone icon will display more details about the person and its phone.
- Double-clicking on it will call the person if its phone is activated
- Right-clicking on it will display the list of possible actions.

- Dragging and dropping it on another person icon will make the dragged user call the dropped user.

Possible actions available through right-click are :

- Call
- Hangup
- Chat
- Intercept call
- Transfer a call to this user
- Cancel a transfer
- Invite to a conference room

The available actions may differ, depending on your current phone situation (available, busy, in a conference room, ...) and on the actions allowed in your CTI profile.

Configuration You can modify the display of contacts within the XLet: Go in the menu XiVO Client -> Configure, tab Functions, sub-tab Contacts. You get two options :

- The maximum number of contacts displayed
- The number of columns used to display the contacts. A value of 0 will automatically display the contacts with the maximum number of columns allowed by the width of the window.

Transfers Many transfers scenarios are supported from the XiVO contact xlet. Blind and attended tranfers can be done by right clicking a contact.

Important: To be able to transfer calls using the XiVO client you have to enable the transfer service from the user configuration (or the queue configuration if used) form in the web-interface.

Attended Transfers

Important: For the Attended Transfer to work properly in all expected cases you must take care of the value of the options below:

Services -→ IPBX -→ Services IPBX -→ Extensions -→ Advanced -→ Parking:

- option *Allow DTMF based transfers when picking up parked call* should be set to `Caller` to be able to initiate an attended transfer for a call picked from a parking,
- option *Allow DTMF based hangups when picking up parked call* should be set to `Caller` to be able to abort an attended transfer picked up from a parking,

---

Usage :

1. Answer an incoming call,
2. Search an user in the Contact xlet,
3. Right clic on the user icon and choose *Attended transfer*,
   (a) If the selected user has also a mobile, you can choose its mobile,
   (b) You can abort the attended transfer by dialing `*0` on your phone (see note below),
   (c) You can finish the attended transfer by hanging up the call,

Other important options to look to are :

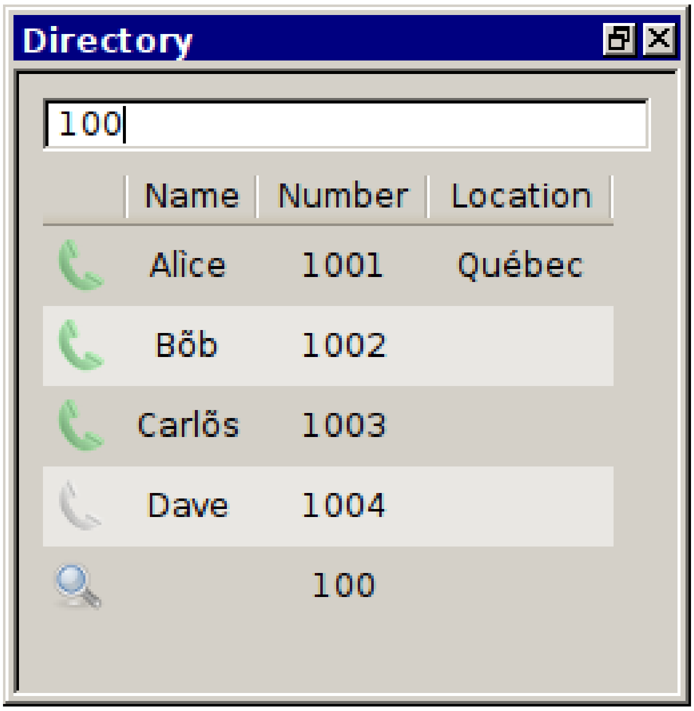
- *Services -→ IPBX -→ Services IPBX -→ Extensions -→ General -> Transfers* : option *Timeout for answer on attended transfer* should be set to a value below the mean ringing time of most users on the XiVO if you want the attented transfer be aborted automatically after this timeout,

- *Services* –→ *IPBX* –→ *Services IPBX* –→ *Extensions* –→ *General* : the option *Hangup* must be set to `*0` if you want to use `*0` to abort attended transfer.

Directory

Overview

The goal of the directory xlet is to allow the user to search through XiVO users, directory entries and arbitrary numbers to be able to transfer calls to these destinations.

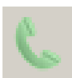
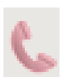


Usage

The list of entries in the xlet is searched using the top field. Entries are filtered by column content. The entry list will initally appear as empty.

If the current search term is a valid number, it will be displayed in the result list with no name to allow transfer to numbers that are not currently in the phonebook or configured on the XiVO.

Legend

- Users available 
- Users ringing 
- Users talking 
- Users 

- Mobile phone 
- External contacts 
- Current search (not a contact) 

Known issues

Typing enter while the focus is on a directory entry will have the same behavior as clicking on the transfer button on the *Switchboard* xlet.

This unexpected behavior will be fixed when we implement other actions to the directory xlet.

Phonebook

Phonebook searches are triggered after the user has entered 3 characters. Results from remote directories will appear after 1 second.

If a directory entry as the same number as a mobile or a phone configured on the XiVO, it's extra columns will be added to the corresponding entry instead of creating a new line in the search result.

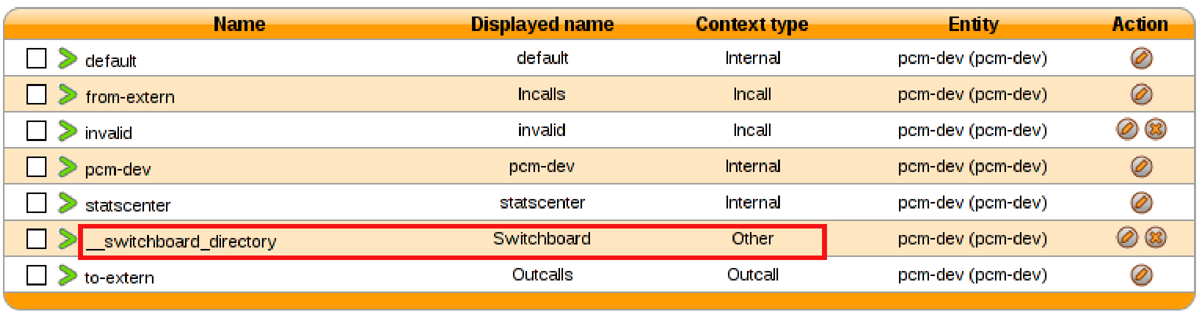
For example:

If *User 1* has number *1000* and is also in a configured LDAP with a location in "Québec", if the display filter contains the *Location* column, the entry for *User 1* will show "Québec" in the *Location* column after the search results are received.

Configuration

Context The directory xlet needs a special context named *__switchboard_directory*. In *Services* → *IPBX* → *IPBX configuration* → *Contexts* add a new context with the followong parameters :

- Name : `__switchboard_directory`
- Type of context : **Other**
- Display name : Switchboard

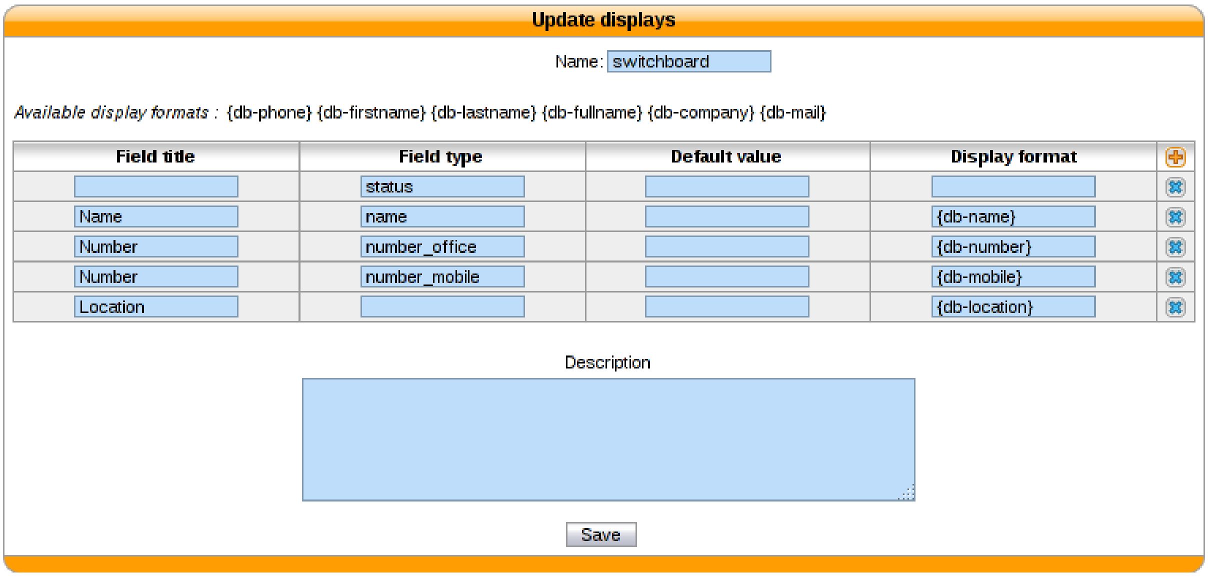


Name	Displayed name	Context type	Entity	Action
default	default	Internal	pcm-dev (pcm-dev)	
from-extern	Incalls	Incall	pcm-dev (pcm-dev)	
invalid	invalid	Incall	pcm-dev (pcm-dev)	
pcm-dev	pcm-dev	Internal	pcm-dev (pcm-dev)	
statscenter	statscenter	Internal	pcm-dev (pcm-dev)	
__switchboard_directory	Switchboard	Other	pcm-dev (pcm-dev)	
to-extern	Outcalls	Outcall	pcm-dev (pcm-dev)	

Display filter A new display filter must be created for the directory xlet.

The following fields must be configured with the correct value for the *Field type* column in order for entries to be displayed in the xlet:

1. *status* is the column that will be used to display the status icon, the title can be empty



2. *name* is displayed in the *Name* column of the xlet
3. *number_office* is displayed in the *Number* column with a phone icon in the xlet
4. *number_mobile* is displayed in the *Number* column with a mobile icon in the xlet
5. *number_...* any other field starting with *number_* will be displayed in the *Number* column of the xlet with a generic directory icon
6. Any other field will be displayed in their own column of the directory xlet

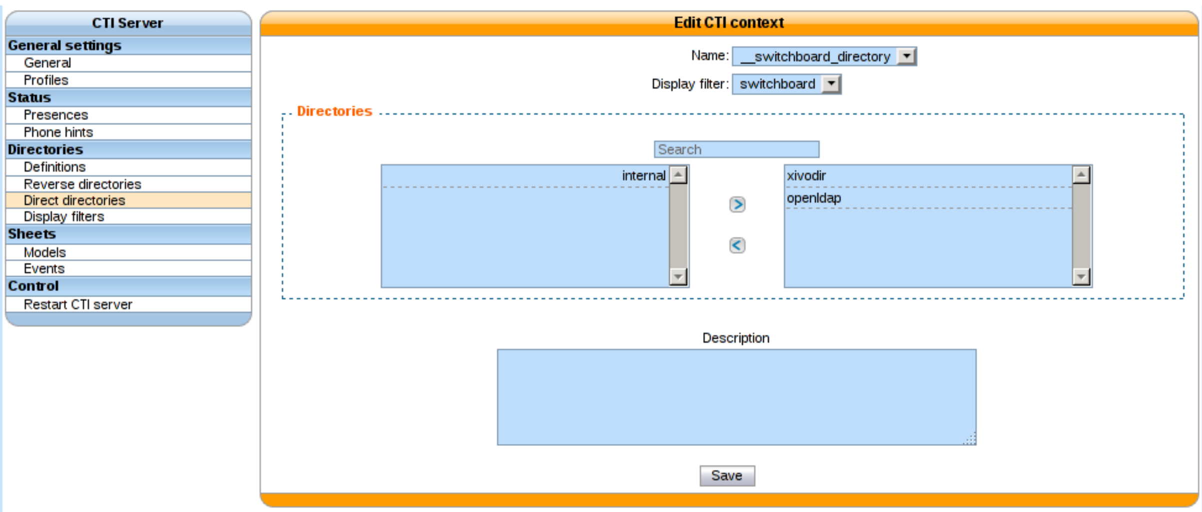
The values in the *Display format* column must contain values that were created in the *Directory definition*.

The title used for the *Number* column is the title of the first field whose type starts with *number_*.

Note: The field title of the first number column will be used for the header title in the xlet.

Warning: Make sure that the fields entered in the display format are also available in the directory definition, otherwise the filter will not return any results

Context and filter association The new *Display filter* has to be assigned to the *__switchboard_directory* context



You can then choose which directories will be searched by the Xlet.

Warning: You must **not select internal** directory, as it is already handled.

LDAP Configuration To search in ldap directories, you must have an LDAP server configured. See *LDAP* for more details.

LDAP filter If you already have an LDAP filter configured for the *Remote directory* Xlet, you can use it.

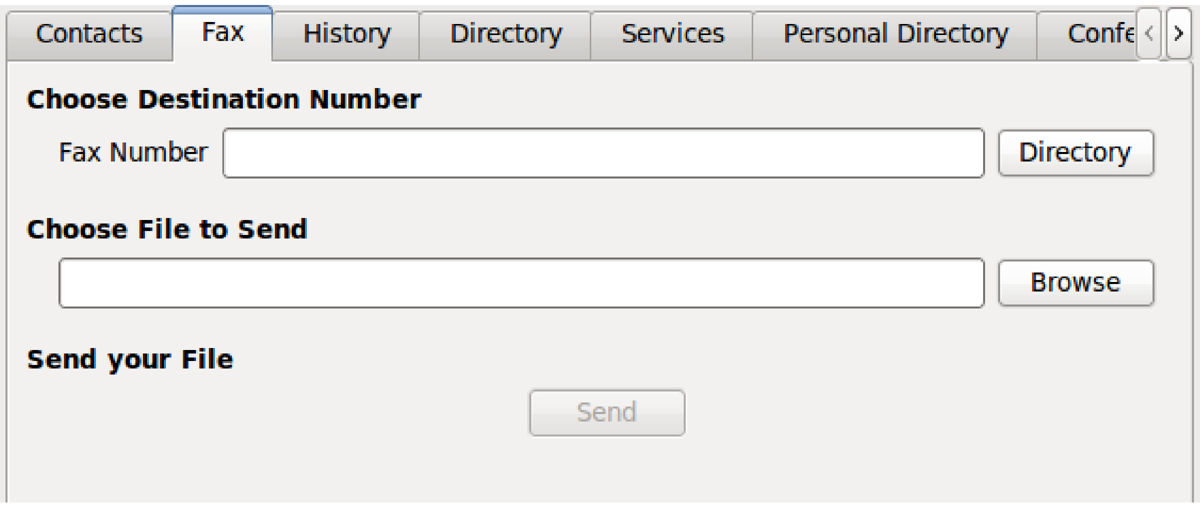
If not, please refer to *Add a LDAP Filter*.

Include the new directory for lookup You must use the new LDAP filter in the Context and filter association step.

Fax Xlet

Overview

The Fax xlet allows the user to send faxes from his XiVO client.



Usage

The *Fax number* field is the fax destination, directory search can be used to find the fax number in available directories. The *Choose a file to send* field is used to select which file you want to send.

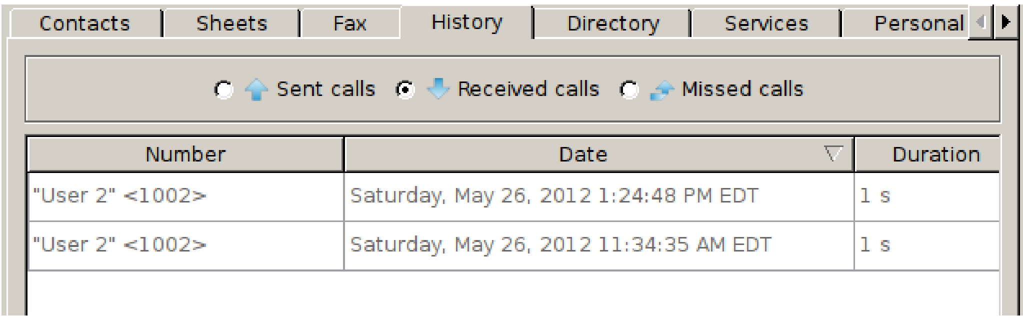
Supported file type

- pdf
- tiff

History Xlet

Overview

The history xlet allow the user to view his sent, received and missed calls.



Usage

The history xlet is made of 3 different call categories, each of them being mutually exclusive ie. a missed call is not in the incoming call list.

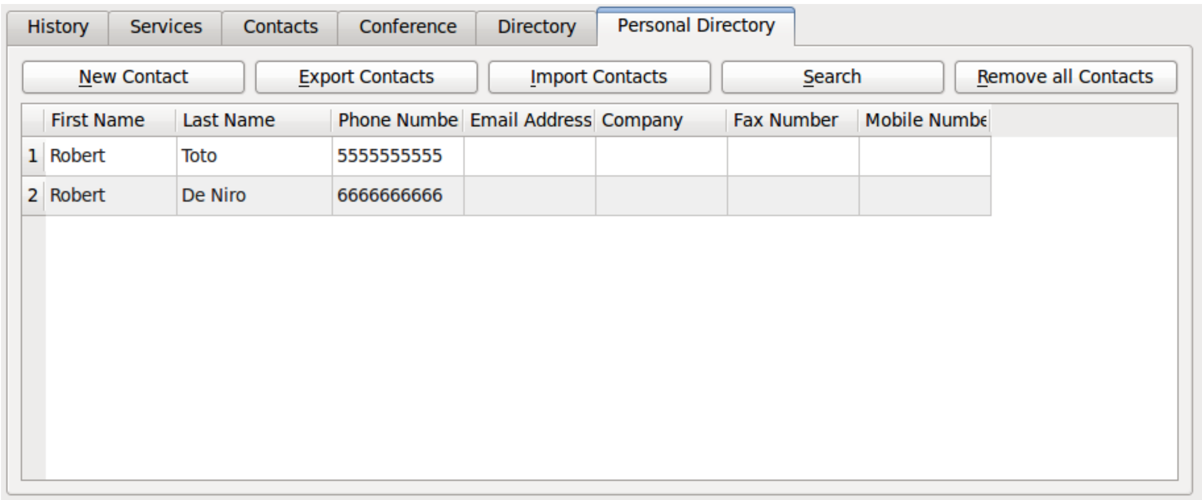
The user can right click on the caller to initiate a new call with a given correspondant.

Warning: The column content is only refreshed when moving from one column to the other.

Local Directory Xlet

Overview

The local directory xlet allow a user to add personnal contacts to the XiVO client.



Usage

The `New Contact` button allow the user to add a new contact to his local contact file.

The `Export Contacts` button allow the user to save a copy of his local contacts in a file in CSV format.

The `Import Contacts` button allow the user to import a CSV file containing new contacts into his local directory.

The `Search` button allow the user to find an occurence of a string in his local directory. Clicking the “Ok” button multiple times in the search dialog will find the next occurence of the searched string.

The `Remove all contacts` button deletes all contacts from the user’s local directory.

File format

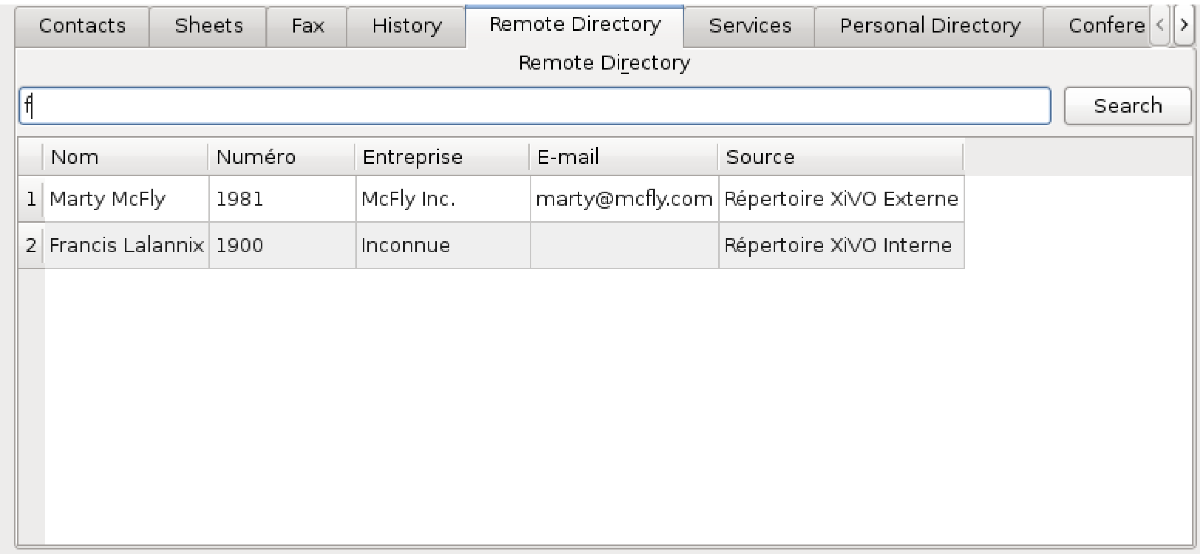
Imported files should have the following structure:

```
firstname,lastname,phonenumber,emailaddress,company,faxnumber,mobilenumber
Robert,Toto,5555555555,my@email,xivo,1234,5551231234
```

Remote Directory Xlet

Overview

The remote directory xlet allows the user to search through the configured directories of the CTI server.



Usage

Type the search term in the search box and click on the search button. If the search box is empty, all phonebook entries will be displayed.

Results can be sorted by clicking on one of the column headers. By default, results are sorted using the first column.

Your sorting preference will be saved and restored every time you reconnect.

> **Warning:** Since phonebook access is managed by context, a user with no line will not receive any result.

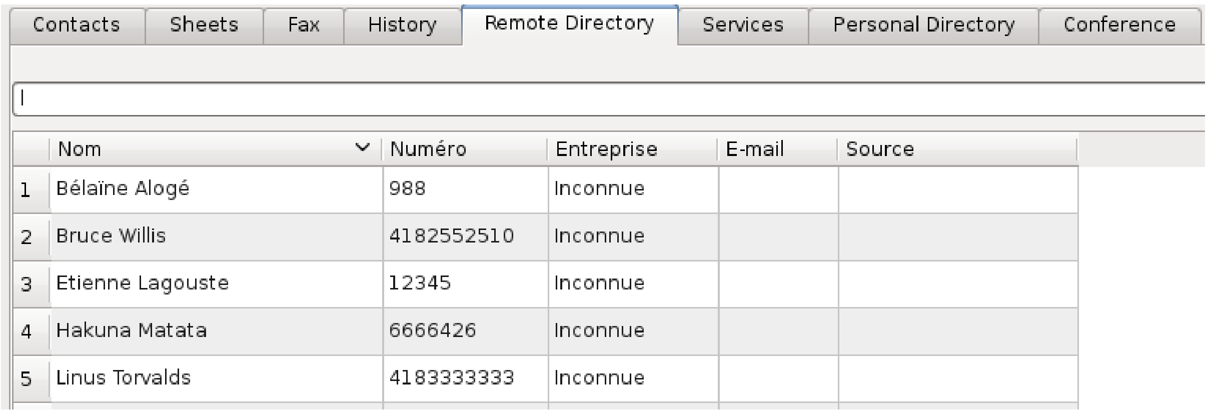
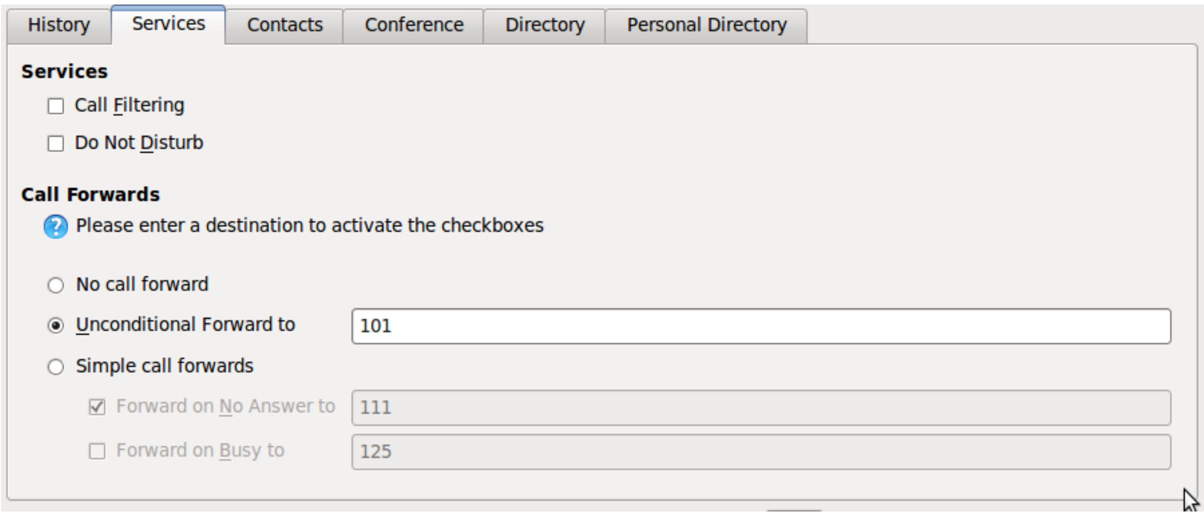


Figure 1.18: Example of contacts sorted by name

Service Xlet

Overview

The service xlet allows the user to enable and disable telephony services such as call forwarding, call filter and do not disturb.



Forward services changes are also reflected in the web interface in the services/ipbx tab under IPBX settings, users. In the user configuration, service tab.

Configuration

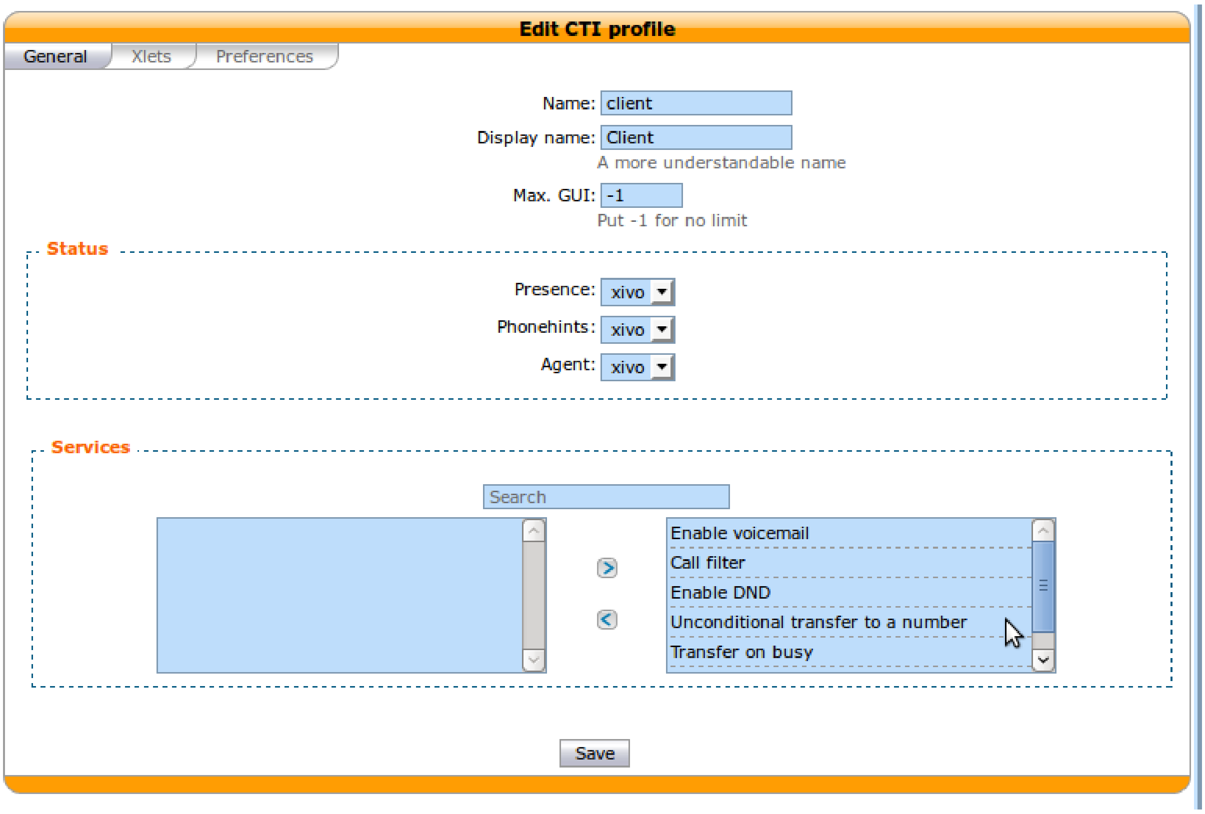
The available service list is configured from the web interface in the services/cti server tab under general settings, profiles.

The right side of the service section contains services that are available to a given profile.

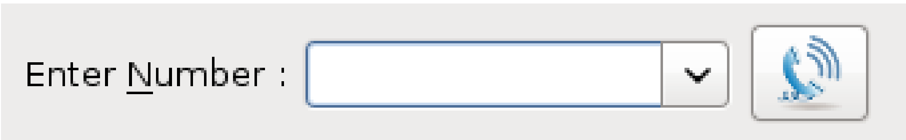
Dial Xlet

Overview

The Dial Xlet allows you to make calls from your computer, via your phone. This means that you can enter the number that you want to dial on your computer, then your phone rings and when you answer it, the called phone



will ring.



Usage

You can enter the number you want to dial in the text box and then click the button or press enter to dial it.

If you dial an invalid extension (a number is an extension), your phone will ring and you will be told that the extension is not valid.

The numbers you dialed are stored in the drop-down list of the text box.

Configuration

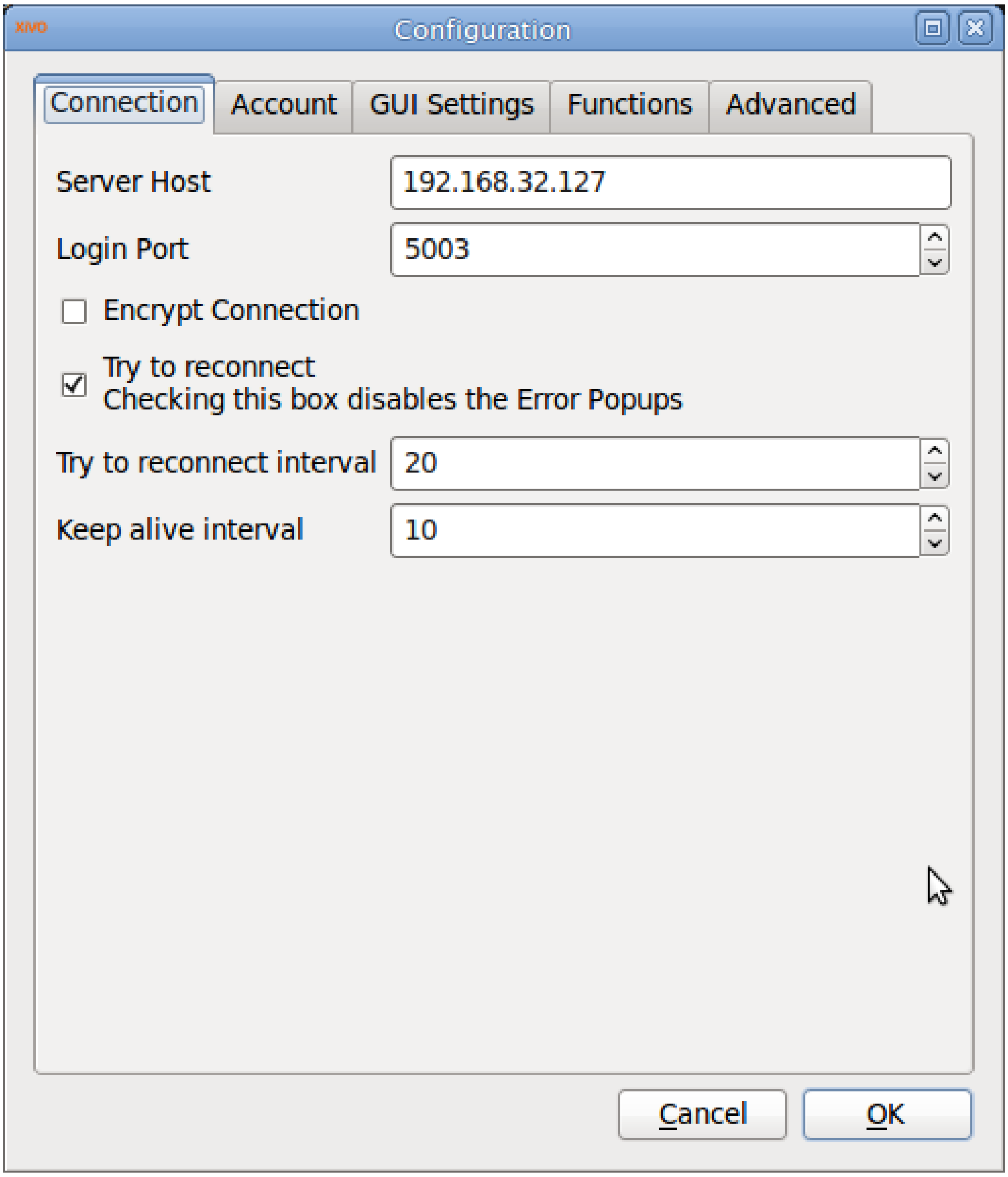
In the menu *XiVO Client* → *Configure* → *Functions* → *Dial*, you can choose how much dialed extensions will be saved between two connections.

In the menu *XiVO Client* → *Configure* → *GUI Settings*, you can enable the integration of the clipboard in the XiVO Client : all text selected in other programs will be automatically pasted in the Dial text box. This feature currently only works on GNU/Linux systems.

1.5.4 Configuration

The XiVO client configuration options can be accessed under *XiVO client* → *configure*.

Connection Configuration



This page allows the user to set his network information to connect to the xivo-ctid server.

- *Server Host* is the IP adress of the server.
- *Login Port* is the port on wich xivo-ctid is listening for connections. (default: 5003)
- *Encrypt Connection* is the option to encrypt messages between the client and the server. (default port 5013)
- *Try to reconnect* will reconnect the client when the connection is dropped.
- *Try to reconnect interval* is the reconnection delay before the auto-reconnection is tryed.
- *Keep alive interval* is the number of seconds between keepalives messages.

1.5.5 Handling callto: and tel: URLs

The XiVO client can handle telephone number links that appear in web pages. The client will automatically dial the number when you click on a link.

Note: You must already be logged in for automatic dialing to work, otherwise the client will simply start up and wait for you to log in.

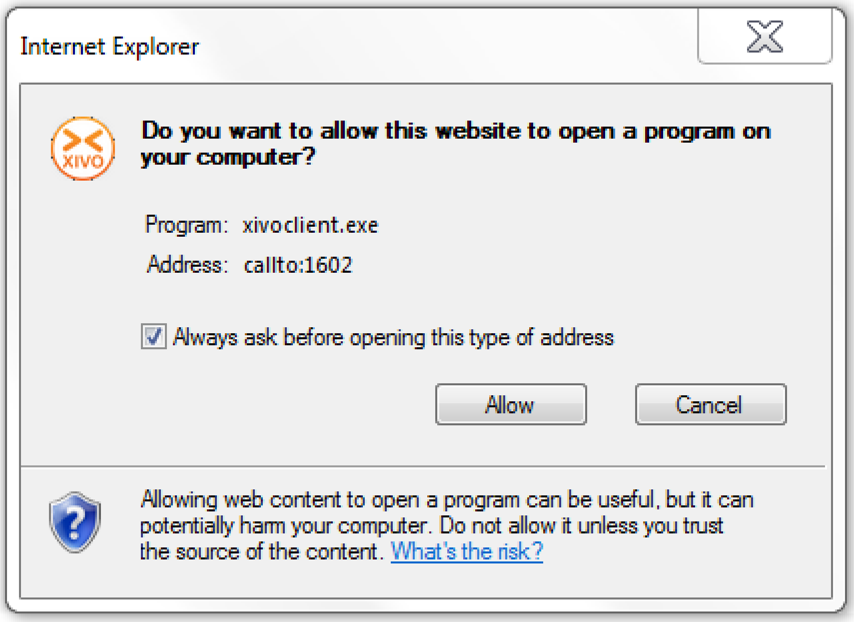
Warning: The option in the XiVO Client *GUI Options → Allow multiple instances of XiVO Client* must be disabled, else you will launch one new XiVO Client with every click.

Mac OS

`callto:` and `tel:` links will work out-of-the-box in Safari and other web browsers after installing the client.

Windows

The following popups might appear When you open a `callto:` or `tel:` link for the first time in Internet Explorer:



Simply click on *allow* to dial the number using the XiVO client.

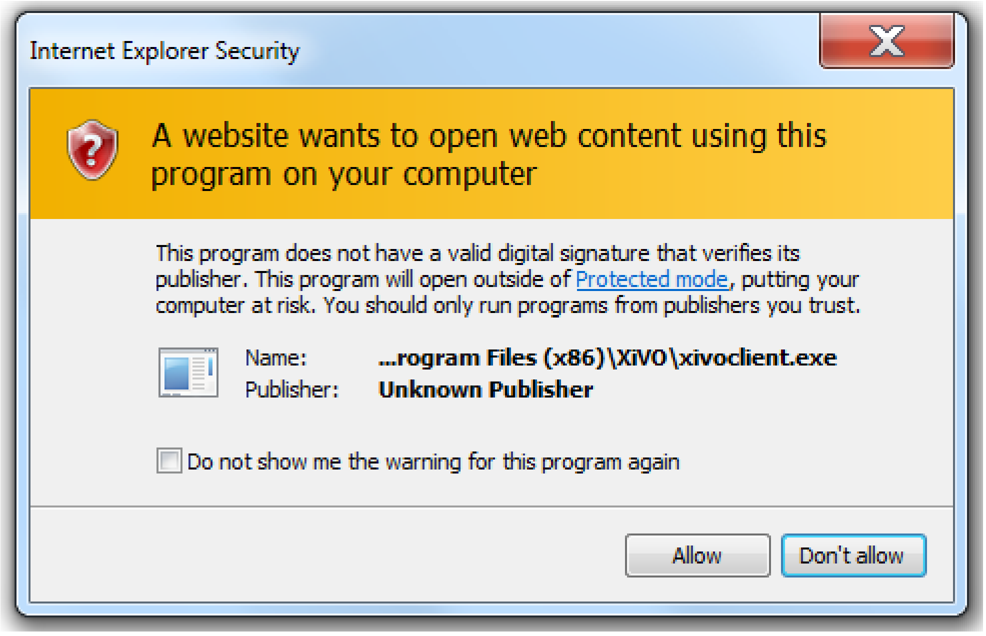
Note: If you do not want these warnings to appear each time, do not forget to check/uncheck the checkbox at the bottom of the popups.

Ubuntu

There is no configuration needed.

GNU/Linux Debian

If the XiVO Client is not listed in the proposition when you open the link, browse your files to find `/usr/bin/xivoclient`.



Manual association in firefox

If, for some reason, firefox does not recognize `callto:` or `tel:` URIs you can manually associate them to the XiVO client using the following steps:

1. Type `about:config` in the URL bar
2. Click the *I'll be careful, I promise !* button to close the warning
3. Right-click anywhere in the list and select *New -> Boolean*
4. Enter `network.protocol-handler.external.callto` as preference name
5. Select `false` as value
6. Repeat steps 3 to 6, but replace `callto` by `tel` at step 4

The next time that you click on a telephone link, firefox will ask you to choose an application. You will then be able to choose the XiVO client for handling telephone numbers.

1.6 System

1.6.1 XiVO service

XiVO has many running services. To restart the whole stack, the *xivo-service* command can be used to make sure the service is restarted in the right order.

Usage

Show all services status:

```
xivo-service status
```

Stop XiVO services:

```
xivo-service stop
```

Start XiVO services:

```
xivo-service start
```

Restart XiVO services:

```
xivo-service restart
```

The commands above will only act upon XiVO services. Appending an argument `all` will also act upon `nginx` and `postgresql`. Example:

```
xivo-service restart all
```

UDP port 5060 will be closed while services are restarting.

1.6.2 Network

You **must** configure your network interfaces directly from the XiVO web interface via the *Configuration* → *Network* → *Interfaces* page.

The Voip interface is used by the DHCP server and the provisioning server.

How-to

You can only have one VoIP interface, which is eth0 by default. This interface is configured during the wizard.

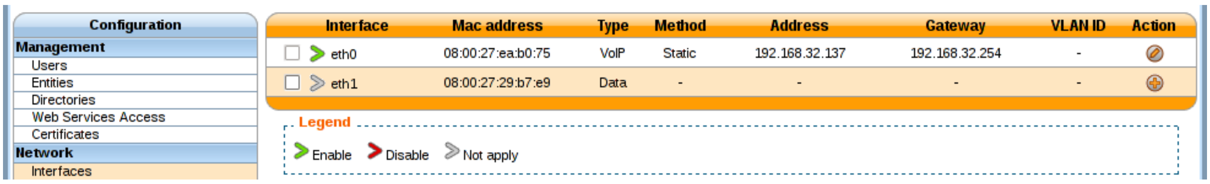
The DHCP server and provisioning server, among other, use information from the VoIP interface in its configuration. For example, the DHCP server will only listen on the VoIP interface per default.

To change this interface, you must either create a new one or edit an existing one and change its type to VoIP. The type of the old interface will automatically be changed to the 'data' type.

Configuring a physical interface

In this example, we'll add and configure the *eth1* network interface on our XiVO.

First, we see there's already an unconfigured network interface named ''eth1" on our system:



Listing the network interfaces

To add and configure it, we click on the small plus button next to it, and we get to this page:

Configure physical interface

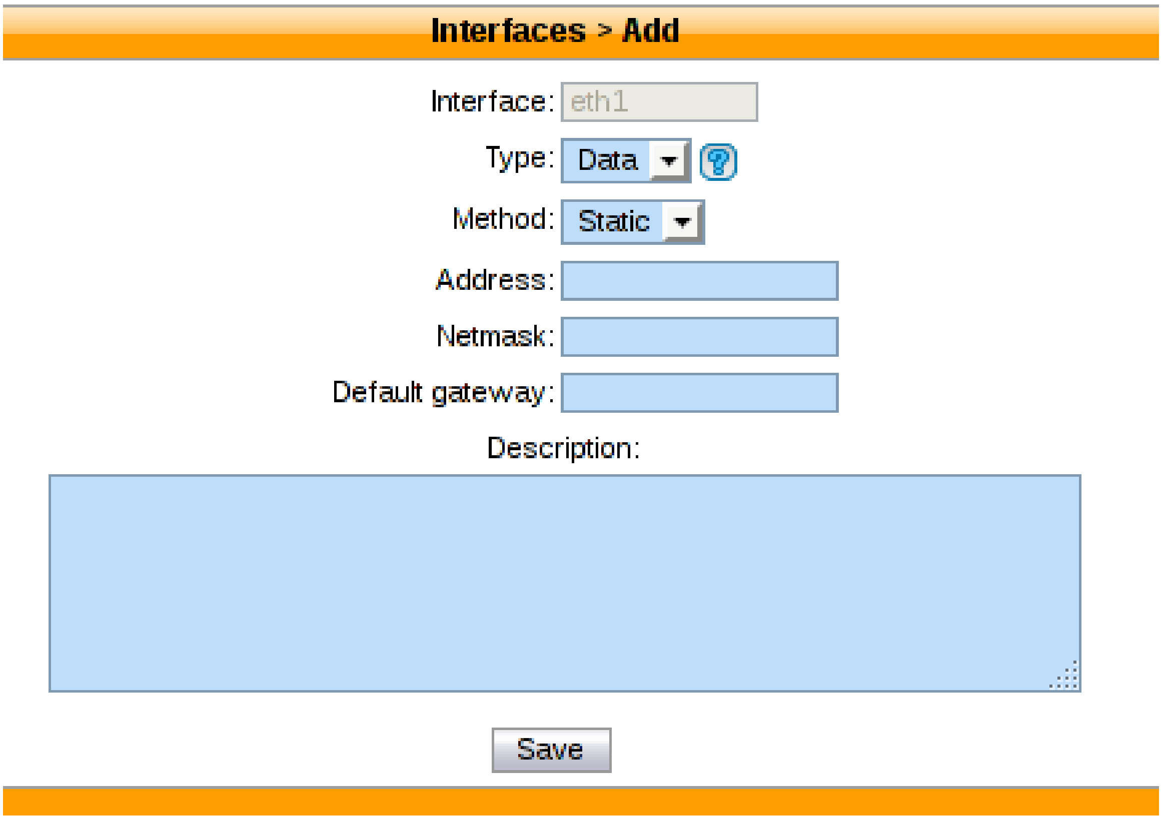
In our case, since we want to configure this interface with static information (i.e. not via DHCP), we fill the following fields:

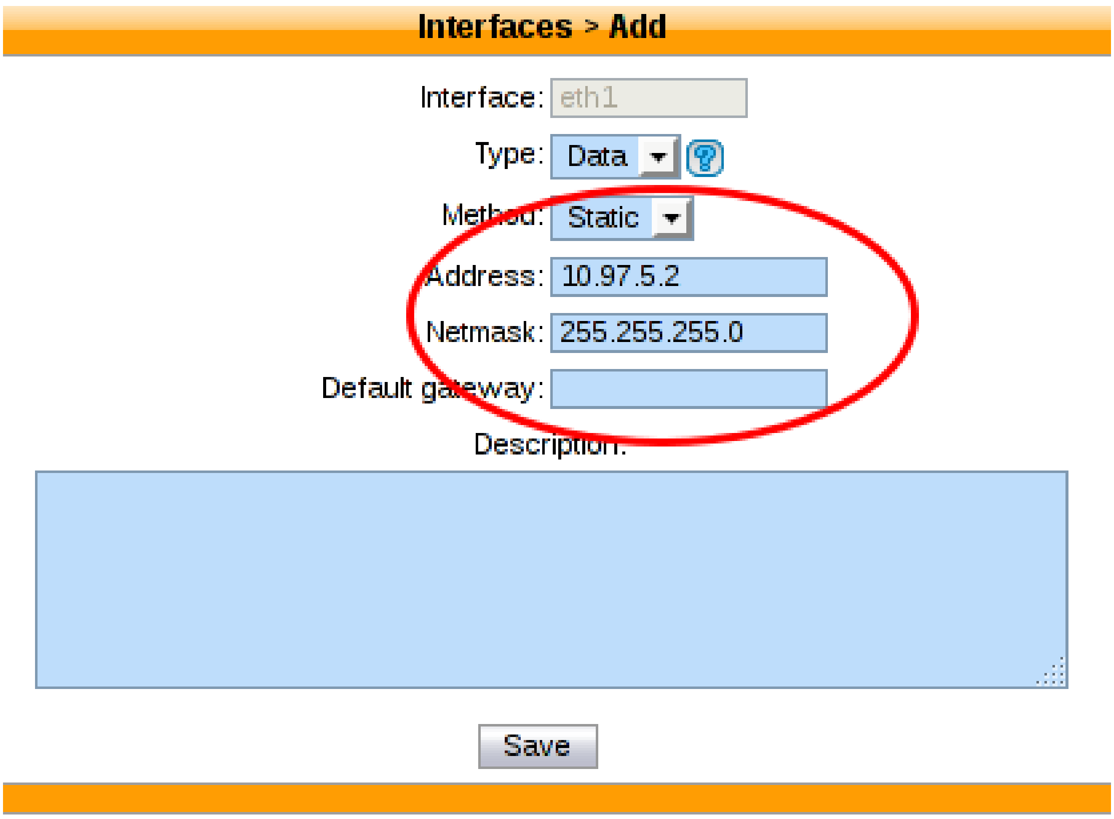
Configure physical interface

Note that since our ''eth0" network interface already has a default gateway, we do not enter information in the ''Default gateway" field for our ''eth1" interface.

Once we click on ''Save'', the XiVO will put the ''Apply network configuration" button in bold.

To reconfigure the given network interface with the new information, you click on it.


Interfaces > Add
Interface: eth1
Type: Data
Method: Static
Address:
Netmask:
Default gateway:
Description:
Save


Interfaces > Add
Interface: eth1
Type: Data
Method: Static
Address: 10.97.5.2
Netmask: 255.255.255.0
Default gateway:
Description:
Save

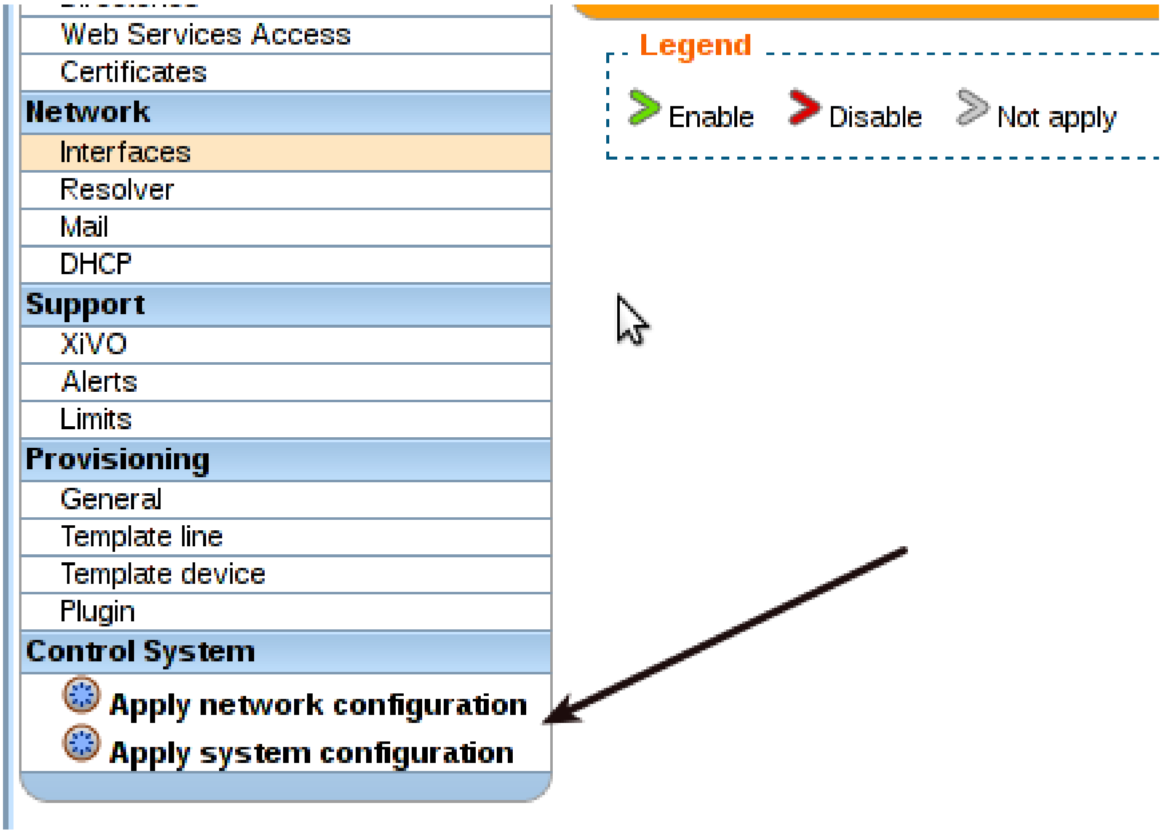
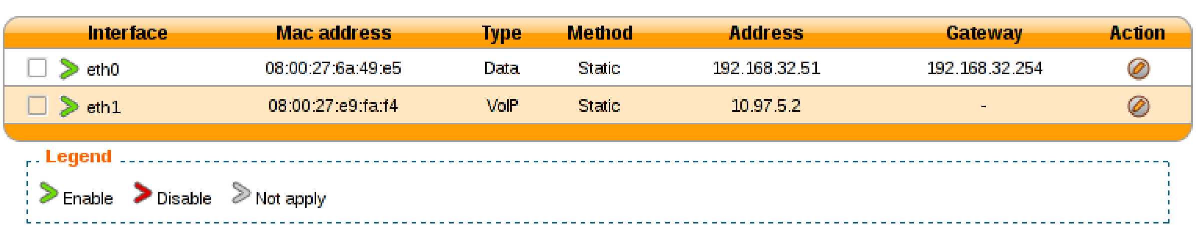


Figure 1.19: Apply after modify interface

Adding a VLAN interface

First, we see there's already a configured network interface on our system:



Interface	Mac address	Type	Method	Address	Gateway	Action
eth0	08:00:27:6a:49:e5	Data	Static	192.168.32.51	192.168.32.254	
eth1	08:00:27:e9:fa:f4	VoIP	Static	10.97.5.2	-	

Listing the network interfaces

To add and configure a new VLAN interface, we click on the small plus button in the top right corner,

and we get to this page:

In our case, since we want to configure this interface with static information:

Click on **Save** list the network interfaces:

- The new virtual interface has been successfully created.

Note: Do not forget after you finish the configuration of the network to apply it with the button: **Apply network configuration**

After applying the network configuration:

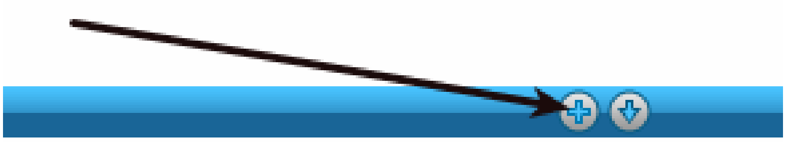


Figure 1.20: Adding button

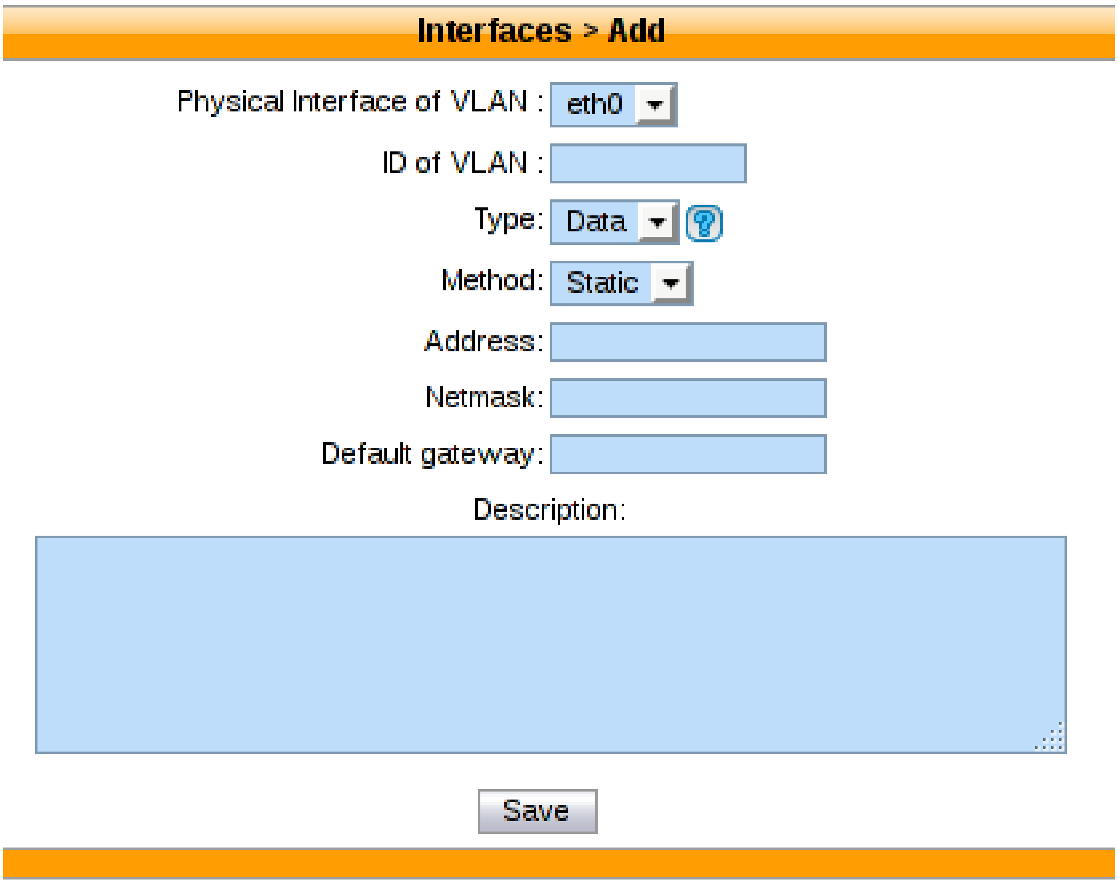


Figure 1.21: Adding a new virtual interface

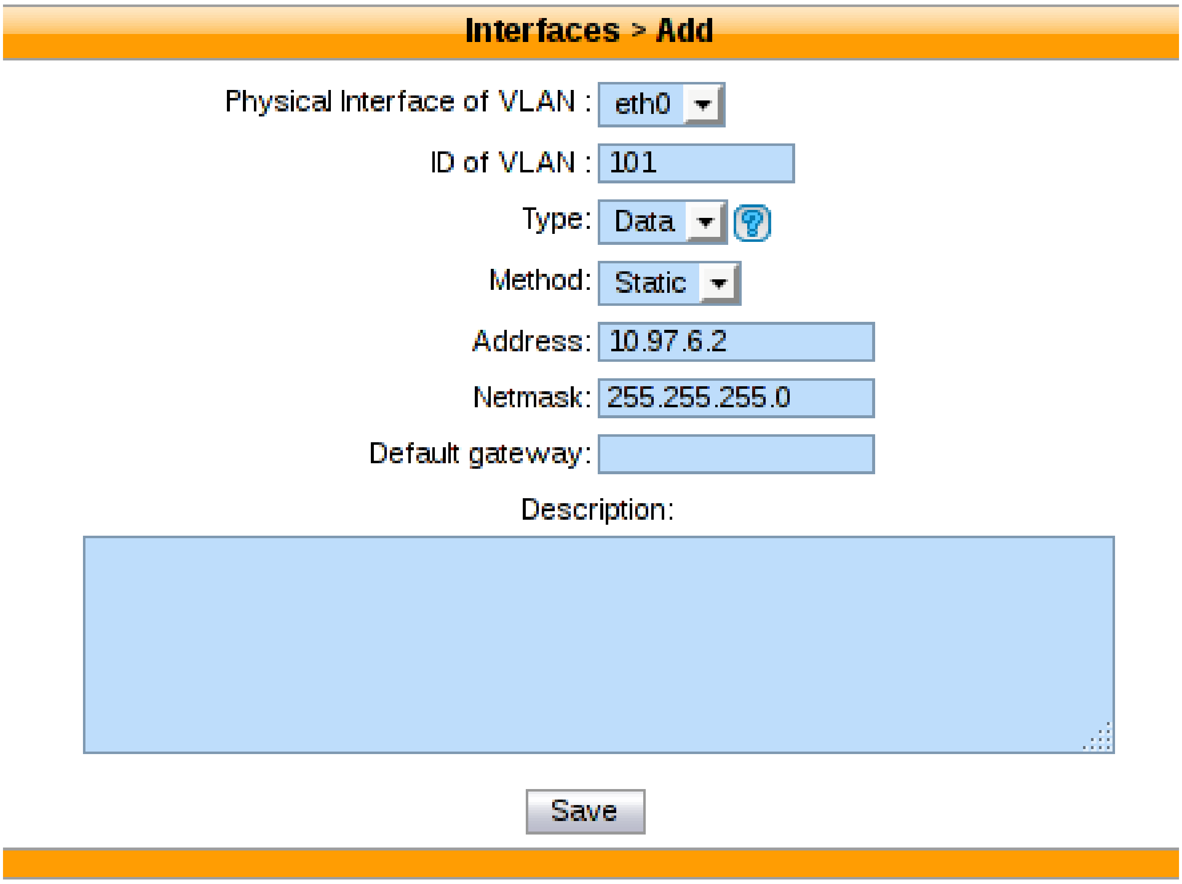


Figure 1.22: Adding a new virtual interface

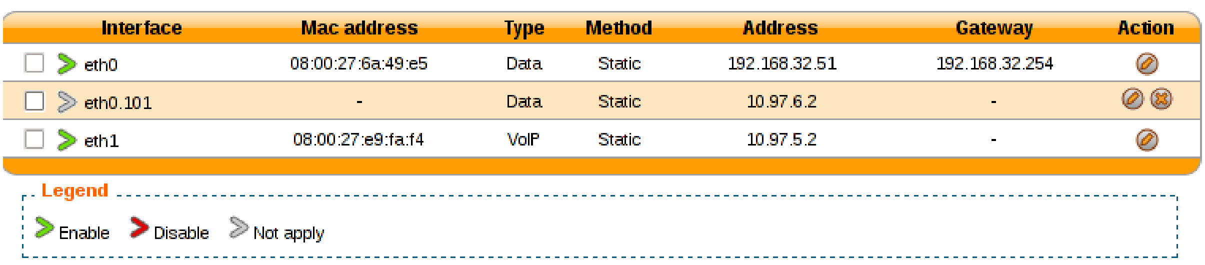


Figure 1.23: Listing the network interfaces

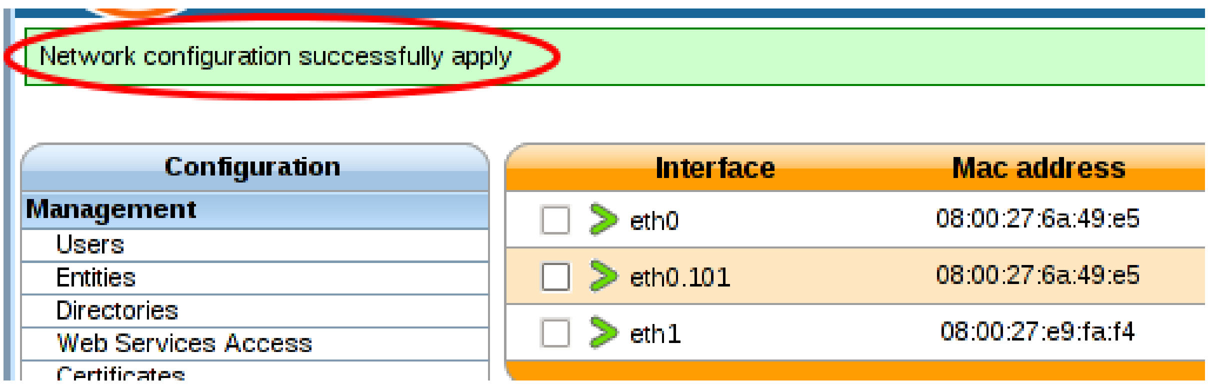


Figure 1.24: Listing the network interfaces

Add static network routes

Static route can't be currently added via the web interface. If you want static routes in your XiVO you should do the following steps described below. It would ensure that your static routes are applied at startup (in fact each time the network interface goes up).

1. Create the file `/etc/network/if-up.d/xivo-routes`:

   ```
   touch /etc/network/if-up.d/xivo-routes
   chmod 755 /etc/network/if-up.d/xivo-routes
   ```

2. Insert the following content:

   ```
   #!/bin/sh

   if [ "${IFACE}" = "<network interface>" ]; then
       ip route add <destination> via <gateway>
       ip route add <destination> via <gateway>
   fi
   ```

3. Fields <network interface>, <destination> and <gateway> should be replaced by your specific configuration. For example, if you want to add a route for 192.168.50.128/25 via 192.168.17.254 which should be added when eth0.100 goes up:

   ```
   #!/bin/sh

   if [ "${IFACE}" = "eth0.100" ]; then
       ip route add 192.168.50.128/25 via 192.168.17.254
   fi
   ```

Note: You need to check which interface goes up to add routes only if the right interface goes up. Otherwise the system will try to set the routes each time any interface goes up.

Change interface MTU

Warning: Changing the MTU is risky. You should know what you are doing.

If you need to change the MTU here is how you should do it:

1. Create the file `/etc/network/if-up.d/xivo-mtu`:

   ```
   touch /etc/network/if-up.d/xivo-mtu
   chmod 755 /etc/network/if-up.d/xivo-mtu
   ```

2. Insert the following content:

   ```
   #!/bin/sh

   # Set MTU per iface
   if [ "${IFACE}" = "<data interface>" ]; then
       ip link set ${IFACE} mtu <data mtu>
   elif [ "${IFACE}" = "<voip interface>" ]; then
       ip link set ${IFACE} mtu <voip mtu>
   fi
   ```

3. Change the *<data interface>* to the name of your interface (e.g. eth0), and the *<data mtu>* to the new MTU (e.g. 1492),

4. Change the *<voip interface>* to the name of your interface (e.g. eth0.10), and the *<voip mtu>* to the new MTU (e.g. 1488)

Note: In the above example you can set a different MTU per interface. If you don't need a per-interface MTU you can simply write:

```
#!/bin/sh

ip link set ${IFACE} mtu <my mtu>
```

1.6.3 Proxy Configuration

If you use XiVO behind an HTTP proxy, you must do a couple of manipulations for it to work correctly.

Global configuration

Some programs are able to use proxy information the `http_proxy` environment variables. You can set and export this variable with:

```
export http_proxy=http://domain\username:password@proxyip:proxyport
```

where

- domain : the user's domain
- username : the username used to login via the proxy
- password : the password used to login via the proxy
- proxyip : the IP of the proxy
- proxyport : the port used by the proxy

If you need to have these settings ready at each connection, you can store them in your `~/.bashrc` file.

If you need to reset the `http_proxy` environment variable, issue the command:

```
unset http_proxy
```

apt

Create the `/etc/apt/apt.conf.d/90proxy` file with the following content:

```
Acquire::http::Proxy "http://domain\username:password@proxyip:proxyport";
```

provd

Proxy information is set via the *Configuration* → *Provisioning* → *General* page.

dhcp-update

This step is needed if you use the DHCP server of the XiVO. Otherwise the DHCP configuration won't be correct.

Proxy information is set via the `/etc/xivo/dhcpd-update.conf` file.

Edit the file and look for the `[proxy]` section.

xivo-fetchfw

This step is not needed if you don't use xivo-fetchfw.

Proxy information is set via the `/etc/xivo/xivo-fetchfw.conf` file.

Edit the file and look for the `[proxy]` section.

External links

- XiVO 1.1 and proxy server

1.6.4 Log Files

Many XiVO services use the syslog's '/var/log/daemon.log' file to log events.

This log file's configuration is located at '/etc/logrotate.d/rsyslog'

The default configuration for all services using this file is the following

- File location: '/var/log/daemon.log'
- Rotation frequence: Weekly
- Number of archived files: 4

agid

The agid log files are sent to the system's syslog.

See log files above for global configuration info.

asterisk

The Asterisk log files are managed by logrotate.

It's configuration files '/etc/logrotate.d/asterisk' and '/etc/asterisk/logger.conf'

The message log level is enabled by default in logger.conf and contains notices, warnings and errors. The full log entry is commented in logger.conf and should only be enabled when verbose debugging is required. Using this option in production would VERY large log files.

Default configuration

- Files location: '/var/log/asterisk/*'
- Number of archived files: 15
- Rotation frequence: Daily

provd

Provd logs are send to the system's syslog.

See log files above for global configuration info.

sysconfd

Sysconfd logs are send to the system's syslog.

See log files above for global configuration info.

web-interface

The web-interface's log file is managed by logrotate.

It's configuration file is '/etc/logrotate.d/xivo-web-interface'

Default configuration

- Rotation frequence: Daily
- Number of archived files: 21
- File location: /var/log/xivo-web-interface/*.log

xivo-confgend

The xivo-confgend daemon output is sent to the file specified with the –logfile parameter when launched with twistd.

The file location can be changed in '/etc/init.d/xivo-confgen'. Search the line begining with 'logfile=/var/log/xivo-confgend.log' and change it to your liking.

Default configuration

- File location: '/var/log/xivo-confgend.log'

xivo-ctid

The xivo-ctid log file is managed by logrotate.

It's configuration file is '/etc/logrotate.d/xivo-ctid'.

Default configuration

- Max log file size: 100M
- Number of archived log files: 15
- Rotation frequence: Daily
- File location: '/var/log/xivo-ctid.pid'

1.6.5 Configuration Files

This section describes some of the XiVO configuration files.

xivo_ring.conf

- Path: `/etc/xivo/asterisk/xivo_ring.conf`
- Purpose: This file can be used to change the ringtone played by the phone depending on the origin of the call.

> **Warning:** Note that this feature has not been tested for all phones and all call flows. This page describes how you can customize this file but does not intend to list all validated call flows or phones.

This file `xivo_ring.conf` consists of :

- profiles of configuration (some examples for different brands are already included: `[aastra]`, `[snom]` etc.)
- one section named `[number]` where you apply the profile to an extension or a context etc.

Here is the process you should follow if you want to use/customize this feature :

1. Create a new profile, e.g.:

   ```
   [myprofile-aastra]
   ```

2. Change the `phonetype` accordingly, in our example:

   ```
   [myprofile-aastra]
   phonetype = aastra
   ```

3. Chose the ringtone for the different type of calls (note that the ringtone names are brand-specific):

   ```
   [myprofile-aastra]
   phonetype = aastra
   intern = <Bellcore-dr1>
   group = <Bellcore-dr2>
   ```

4. Apply your profile, in the section `[number]`

- to a given list of extensions (e.g. 1001 and 1002):

  ```
  1001@default = myprofile-aastra
  1002@default = myprofile-aastra
  ```

- or to a whole context (e.g. default):

  ```
  @default = myprofile-aastra
  ```

5. Restart `xivo-agid` service:

   ```
   service xivo-agid restart
   ```

ipbx.ini

- Path: `/etc/xivo/web-interface/ipbx.ini`
- Purpose: This file specifies various configuration options and paths related to Asterisk and used by the web interface.

Here is a partial glimpse of what can be configured in file `ipbx.ini` :

1. Enable/Disable modification of SIP line username and password:

   ```
   [user]
   readonly-idpwd = "true"
   ```

   When editing a SIP line, the username and password fields cannot be modified via the web interface. Set this option to false to enable the modification of both fields. This option is set to “true” by default.

> **Warning:** This feature is not fully tested. It should be used only when absolutely necessary and with great care.

1.6.6 Backup

Periodic backup

A backup of the database and the data are launched every day with a logrotate task. It is run at 06:25 a.m. and backups are kept for 7 days.

Logrotate task:

```
/etc/logrotate.d/xivo-backup
```

Logrotate cron:

```
/etc/cron.daily/logrotate
```

Retrieve the backup

You can retrieve the backup from the web-interface in *Services* → *IPBX* → *IPBX Configuration* → *Backup Files* page.

Otherwise with a shell access you can retrieve them in `/var/backups/xivo/`. In this directory you will find `db.tgz` and `data.tgz` files for the database and data backup.

Backup script:

```
/usr/sbin/xivo-backup
```

Backup location:

```
/var/backups/xivo/
```

What is actually backuped ?

Data

Here is the list of the folder and files backuped:

- `/etc/asterisk`
- `/etc/dahdi`
- `/etc/dhcp`
- `/etc/hostname`
- `/etc/hosts`
- `/etc/network/interfaces`
- `/etc/ntp.conf`
- `/etc/xivo`
- `/etc/resolv.conf`
- `/etc/wanpipe`
- `/var/lib/asterisk/`
- `/var/lib/xivo/`
- `/var/lib/xivo-provd`
- `/var/log/asterisk/`
- `/var/spool/asterisk/`

The following files/folders are excluded from this backup:

- folders:
  - `/var/spool/asterisk/monitor`
  - `/var/spool/asterisk/meetme`
- log files, coredump files,
- audio recordings,
- and, files greater than 10 MiB or folders containing more than 100 files if they belong to one of these folders:
  - `/var/lib/xivo/sounds`
  - `/var/lib/asterisk/sounds/custom`

– /var/lib/asterisk/moh
– /var/spool/asterisk/voicemail
– /var/spool/asterisk/monitor

Database

Creating a database backup file manually

Warning: A backup file may take a lot of space on the disk. You should check the free space on the partition before doing it.

You can manually create a *database* backup file named `db-manual.tgz` in `/var/tmp` by issuing the following commands:

```
xivo-backup db /var/tmp/db-manual
```

Creating a data backup file manually

Warning: A backup file may take a lot of space on the disk. You should check the free space on the partition before doing it.

You can manually create a *data* backup file named `data-manual.tgz` in `/var/tmp` by issuing the following commands:

```
xivo-backup data /var/tmp/data-manual
```

1.6.7 Restore

Introduction

A backup of both the configuration files and the database used by a XiVO installation is done automatically every day. These backups are created in the `/var/backups/xivo` directory and are kept for 7 days.

Before Restoring the System

Warning: Before restoring a XiVO on a fresh install you have to setup XiVO using the wizard (see *Running the Wizard* section).

Stop monit and all the xivo services:

```
xivo-service stop
```

Restoring System Files

System files are stored in the data.tgz file located in the `/var/backups/xivo` directory.

This file contains for example, voicemail files, musics, voice guides, phone sets firmwares, provisioning server configuration database.

To restore the file

```
tar xvfp /var/backups/xivo/data.tgz -C /
```

Restoring the Database

> **Warning:**
> - This will destroy all the current data in your database.
> - You have to check the free space on your system partition before extracting the backups.

Database backups are created as `db.tgz` files in the `/var/backups/xivo` directory. These tarballs contains a dump of the database used in XiVO.

In this example, we'll restore the database from a backup file named `db.tgz` placed in the home directory of root.

Then, extract the content of the `db.tgz` file into the `/var/tmp` directory and go inside the newly created directory:

```
tar xvf db.tgz -C /var/tmp
cd /var/tmp/pg-backup
```

Drop the asterisk database and restore it with the one from the backup:

```
sudo -u postgres dropdb asterisk
sudo -u postgres pg_restore -C -d postgres asterisk-*.dump
```

Restoring and Keeping System Configuration

System configuration like network interfaces is stored in the database. It is possible to keep this configuration and only restore xivo data.

Rename the asterisk database to asterisk_previous:

```
sudo -u postgres psql -c 'ALTER DATABASE asterisk RENAME TO asterisk_previous'
```

Restore the asterisk database from the backup:

```
sudo -u postgres pg_restore -C -d postgres asterisk-*.dump
```

Restore the system configuration tables from the asterisk_previous database:

```
sudo -u postgres pg_dump -c -t dhcp -t netiface -t resolvconf asterisk_previous | sudo -u postgre
```

Drop the asterisk_previous database:

```
sudo -u postgres dropdb asterisk_previous
```

> **Warning:** Restoring the data.tgz file restore also system files as host hostname network interfaces etc... You will need to reapply network configuration if you restore the data.tgz file

After Restoring The System

Restart the services you stopped at the first step:

```
xivo-service start
```

You may also reboot the system.

1.6.8 DHCP Server

XiVO includes a DHCP server that must be used to address telephony devices (*Basic Configuration*) of the VOIP subnet. This section describes how to configure DHCP server for other subnets or with advanced options.

Activation of DHCP server

DHCP Server can be activated through the XiVO Web Interface *Configuration → Network → DHCP* :

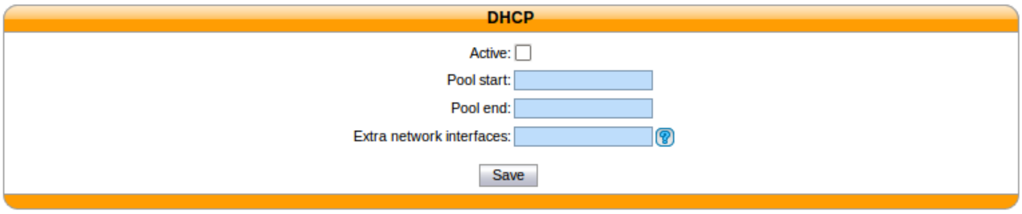


Figure 1.25: *Configuration → Network → DHCP*

By default, it will only answer to DHCP requests coming from the VoIP subnet (defined in the *Configuration → Network → Interfaces* section). If you need to activate DHCP server on an other interface, you have to fill the *Extra network interfaces* field with, for example : `eth0`

After saving your modifications, you need to click on *Apply system configuration* for them to be applied.

Change default gateway for DHCP

By default, the XiVO DHCP server gives the XiVO IP address in the router option. To change this you must create a custom-template:

1. Create a custom template for the `dhcpd_subnet.conf.head` file:

```
mkdir -p /etc/xivo/custom-templates/dhcp/etc/dhcp/
cd /etc/xivo/custom-templates/dhcp/etc/dhcp/
cp /usr/share/xivo-config/templates/dhcp/etc/dhcp/dhcpd_subnet.conf.head .
```

2. Edit the custom template:

```
vim dhcpd_subnet.conf.head
```

3. In the file, replace the string `#XIVO_NET4_IP#` by the router of your VoIP network, for example:

```
option routers 192.168.2.254;
```

4. Re-generate the dhcp configuration:

```
xivo-update-config
```

DHCP server should have been restarted and should now give the new router option.

Configuring DHCP server to serve unknown hosts

By default, the XiVO DHCP server serves only known hosts. That is:

- either hosts which MAC address prefix (the OUI) is known
- or hosts which Vendor Identifier is known

Known OUIs and Vendor Class Identifiers are declared in `/etc/dhcpd/dhcpd_update/*` files.

If you want your XiVO DHCP server to serve also unknown hosts (like PCs) follow these instructions:

1. Create a custom template for the `dhcpd_subnet.conf.tail` file:

   ```
   mkdir -p /etc/xivo/custom-templates/dhcp/etc/dhcp/
   cd /etc/xivo/custom-templates/dhcp/etc/dhcp/
   cp /usr/share/xivo-config/templates/dhcp/etc/dhcp/dhcpd_subnet.conf.tail .
   ```

2. Edit the custom template:

   ```
   vim dhcpd_subnet.conf.tail
   ```

3. And add the following line at the head of the file:

   ```
   allow unknown-clients;
   ```

4. Re-generate the dhcp configuration:

   ```
   xivo-update-config
   ```

DHCP server should have been restarted and should now serve all network equipments.

DHCP-Relay

If your telephony devices aren't located on the same site and the same broadcast domain as the XiVO DHCP server, you will have to add the option *DHCP Relay* to the site's router. This parameter will permit the DHCP requests from distant devices to be transmitted to the IP address you specify as DHCP Relay.

Warning: Please make sure that the IP address used as DHCP Relay is one of the XiVO interface, and that this interface is configured to listen to DHCP requests (as decribed in previous part). Also verify that routing is configured between the distant router and the choosen interface, otherwise DHCP requests will never reach the XiVO server.

Configuring DHCP server for other subnets

This section describes how to configure XiVO to serve other subnets that the VOIP subnet. As you can't use the Web Interface to declare other subnets (for example to address DATA subnet, or a VOIP subnet that isn't on the same site that XiVO server), you'll have to do the following configuration in Command Line Interface.

Creating "extra subnet" configuration files

First thing to do is to create a directory and to copy into it the configuration files:

```
mkdir /etc/dhcp/dhcpd_sites/
cp /etc/dhcp/dhcpd_subnet.conf /etc/dhcp/dhcpd_sites/dhcpd_siteXXX.conf
cp /etc/dhcp/dhcpd_subnet.conf /etc/dhcp/dhcpd_sites/dhcpd_lanDATA.conf
```

Note: In this case we'll create 2 files for 2 differents subnets. You can change the name of the files, and create as many files as you want in the folder `/etc/dhcp/dhcpd_sites/`. Just adapt this procedure by changing the name of the file in the different links.

After creating one or several files in `/etc/dhcp/dhcpd_sites/`, you have to edit the file `/etc/dhcp/dhcpd_extra.conf` and add the according include statement like:

```
include "/etc/dhcp/dhcpd_sites/dhcpd_siteXXX.conf";
include "/etc/dhcp/dhcpd_sites/dhcpd_lanDATA.conf";
```

Adjusting Options of the DHCP server

Once you have created the subnet in the DHCP server, you must edit each configuration file (the files in `/etc/dhcp/dhcpd_sites/`) and modify the different parameters. In section **subnet**, write the IP subnet and change the following options (underlined fields in the example):

```
subnet 172.30.8.0 netmask 255.255.255.0 {
```

- subnet-mask:

  ```
  option subnet-mask 255.255.255.0;
  ```

- broadcast-address:

  ```
  option broadcast-address 172.30.8.255;
  ```

- routers (specify the IP address of the router that will be the default gateway of the site):

  ```
  option routers 172.30.8.1;
  ```

In section **pool**, modify the options:

```
pool {
```

- log (add the name of the site or of the subnet):

  ```
  log(concat("[", binary-to-ascii(16, 8, ":", hardware), "] POOL VoIP Site XXX"));
  ```

- range (it will define the range of IP address the DHCP server can use to address the devices of that subnet):

  ```
  range 172.30.8.10 172.30.8.200;
  ```

> **Warning:** XiVO only answers to DHCP requests from *supported devices*. In case of you need to address other equipment, use the option *allow unknown-clients;* in the `/etc/dhcp/dhcpd_sites/` files

At this point, you can apply the changes of the DHCP server with the command:

```
/etc/init.d/isc-dhcp-server restart
```

After that, XiVO will start to serve the DHCP requests of the devices located on other site or other subnet than the VOIP subnet. You will see in `/var/log/daemon.log` all the DHCP requests receided and how they are handled by XiVO.

1.6.9 NTP

XiVO has a NTP server, that must be synchronized to a reference server. This can be a public one or customized for specific target networking architecture. XiVO's NTP server is used by default as NTP server for the devices time reference.

Usage

Show NTP service status:

```
/etc/init.d/ntp status
```

Stop NTP service:

```
/etc/init.d/ntp stop
```

Start NTP service:

```
/etc/init.d/ntp start
```

Restart NTP service:

```
/etc/init.d/ntp restart
```

Show NTP synchronization status:

```
ntpq -p
```

Configuring NTP service

1. Edit `/etc/ntp.conf`
2. Give your NTP reference servers:

   ```
   server 192.168.0.1                           # LAN existing NTP Server
   server 0.debian.pool.ntp.org iburst dynamic  # default in ntp.conf
   server 1.debian.pool.ntp.org iburst dynamic  # default in ntp.conf
   ```

3. If no reference server to synchronize to, add this to synchronize locally:

   ```
   server 127.127.1.0             # local clock (LCL)
   fudge 127.127.1.0 stratum 10   # LCL is not very reliable
   ```

4. Restart NTP service
5. Check NTP synchronization status.

> **Warning:** If #5 shows that NTP doesn't use NTP configuration in `/etc/ntp.conf`, maybe have you done a `dhclient` for one of your network interface and the dhcp server that gave the IP address also gave a NTP server address. Thus you might check if the file `/var/lib/ntp/ntp.conf.dhcp` exists, if yes, this is used for NTP configuration prior to `/etc/ntp.conf`. Remove it and restart NTP, check NTP synchronization status, then it should work.

1.6.10 Mail

This section describes how to configure the mail server shipped with XiVO (Postfix) and the way XiVO handles mails.

In *Configuration* → *Network* → *Mail*, the following options can be configured:

- *Domain Name messaging* : the server's displayed domain. Will appear in "Received" mail headers.
- *Source address of the server* : domain part of headers "Return-Path" and "From".
- *Relay SMTP* and *FallBack relay SMTP* : relay mail servers.
- *Rewriting shipping addresses* : Canonical address Rewriting. See Postfix canonical documentation for more info.

> **Warning:** Postfix, the mail server shipped with XiVO, should be stopped on an installed XiVO with no valid and reachable DNS servers configured. If Postfix is not stopped, messages will bounce in queues and could end up affecting core pbx features.

If you need to disable Postfix here is how you should do it:

```
/etc/init.d/postfix stop
insserv -r postfix
```

If you ever need to enable Postfix again:

```
insserv postfix
/etc/init.d/postfix start
```

Alternatively, you can empty Postfix's queues by issuing the following commands on the XiVO server:

```
postsuper -d ALL
```

1.7 Ecosystem

1.7.1 Devices

Supported device (Official support)

XiVO provides official support for the following phones. These phones will be supported across upgrades and phone features are guaranteed to be supported on the latest version.

`xivo-provd` plugins for these devices can be installed from the *"officially supported devices" repository*.

> **Warning:** Funckeys work using the extensions in *Services* → *Extensions*. It is important to enable the function keys you want to use.

> **Warning:** The enable transfer option in the user configuration services tab must be enabled to use transfer function keys.

FK = Funckey

HK = HardKey

Y = Supported

MN = Menu

N = Not supported

NT = Not tested

NYT = Not yet tested

SK = SoftKey

Aastra

6700i series:

	6731i	6735i	6737i	6739i	6755i	6757i
Provisioning	Y	Y	Y	Y	Y	Y
H-A	Y	Y	Y	Y	Y	Y
Directory XIVO	Y	Y	Y	Y	Y	Y
Funckeys	8	26	30	55	26	30
	Supported programmable keys					
User with supervision function	Y	Y	Y	Y	Y	Y
Group	Y	Y	Y	Y	Y	Y
Queue	Y	Y	Y	Y	Y	Y
Conference Room with supervision function	Y	Y	Y	Y	Y	Y
General Functions						
Online call recording	N	N	N	N	N	N
Phone status	Y	Y	Y	Y	Y	Y

Continued on next page

Table 1.1 – continued from previous page

	6731i	6735i	6737i	6739i	6755i	6757i
Sound recording	Y	Y	Y	Y	Y	Y
Call recording	Y	Y	Y	Y	Y	Y
Incoming call filtering	Y	Y	Y	Y	Y	Y
Do not disturb	Y	Y	Y	Y	Y	Y
Group interception	Y	Y	Y	Y	Y	Y
Listen to online calls	Y	Y	Y	Y	Y	Y
Directory access	Y	Y	Y	Y	Y	Y
Filtering Boss - Secretary	Y	Y	Y	Y	Y	Y
Transfers Functions						
Blind transfer	HK	Y	Y	HK	Y	Y
Indirect transfer	HK	Y	Y	HK	Y	Y
Forwards Functions						
Disable all forwarding	Y	Y	Y	Y	Y	Y
Enable/Disable forwarding on no answer	Y	Y	Y	Y	Y	Y
Enable/Disable forwarding on busy	Y	Y	Y	Y	Y	Y
Enable/Disable forwarding unconditional	Y	Y	Y	Y	Y	Y
Voicemail Functions						
Enable voicemail with supervision function	Y	Y	Y	Y	Y	Y
Activate voicemail	Y	Y	Y	Y	Y	Y
Reach the voicemail	Y	Y	Y	HK	Y	Y
Delete messages from voicemail	Y	Y	Y	Y	Y	Y
Agent Functions						
Connect/Disconnect a static agent	Y	Y	Y	Y	Y	Y
Connect a static agent	Y	Y	Y	Y	Y	Y
Disconnect a static agent	Y	Y	Y	Y	Y	Y
Parking Functions						
Parking	Y	Y	Y	Y	Y	Y
Parking position	Y	Y	Y	Y	Y	Y
Paging Functions						
Paging	Y	Y	Y	Y	Y	Y

The M670i and M675i expansion modules are supported.

DECT Infrastructure:

	RFP35	RFP36
Provisioning [5]	N [6]	N [4]
H-A [7]	N	N
Directory XIVO	N	N
Funckeys [8]	0	0

Cisco

ATAs:

[1] `Tested` means the device has been tested by the XiVO development team and that the developers have access to this device.

[2] These devices are marked as `Not Tested` because other similar models using the same firmware have been tested instead. If these devices ever present any bugs, they will be troubleshooted by the XiVO support team.

[3] `XiVO HA` means the device is confirmed to work with *XiVO HA*.

[4] `Fkeys` is the number of programmable function keys that you can configure from the XiVO web interface. It is not necessarily the same as the number of physical function keys the device has (for example, a 6757i has 12 physical keys but you can configure 30 function keys because of the page system).

[5] `Tested` means the device has been tested by the XiVO development team and that the developers have access to this device.

[6] These devices are marked as `Not Tested` because other similar models using the same firmware have been tested instead. If these devices ever present any bugs, they will be troubleshooted by the XiVO support team.

[7] `XiVO HA` means the device is confirmed to work with *XiVO HA*.

[8] `Fkeys` is the number of programmable function keys that you can configure from the XiVO web interface. It is not necessarily the same as the number of physical function keys the device has (for example, a 6757i has 12 physical keys but you can configure 30 function keys because of the page system).

	SPA122	SPA3102	SPA8000
Provisioning [1]	N [4]	Y	Y
H-A [3]	N	N	N
Directory XIVO	N	N	N
Funckeys [2]	0	0	0

Note: For best results, activate *DHCP Integration* on your XiVO.

Note: These devices can be used to connect Faxes. For better success with faxes some parameters must be changed. You can read the *Using analog gateways* section.

Note: If you want to manually resynchronize the configuration from the ATA device you should use the following url:

```
http://ATA_IP/admin/resync?http://XIVO_IP:8667/CONF_FILE
```

where :

- *ATA_IP* is the IP address of the ATA,
- *XIVO_IP* is the IP address of your XiVO,
- *CONF_FILE* is one of `spa3102.cfg`, `spa8000.cfg`

Warning: SCCP phones are supported, but limited to the features supported in XIVO's SCCP implementation.

Warning: Access to CISCO firmware updates requires a CISCO account with sufficient privileges. The account requires paying for the service and remains under the responsibility of the client or partner. Avencall is not responsible for these firmwares and does not offer any updates.

Cisco 7900 series (*SCCP* mode only):

	7905G	7906G	7911G	7912G	7920	7921G	7940G	7941
Provisioning [1]	N [4]	N	N [4]	Y	Y	Y	Y	Y
H-A [3]	N	Y	Y	Y	NT	NT	Y	Y
Directory XIVO	N	N	N	FK	N	N	FK	FK
Funckeys [2]	N	4	4	4	0	0	1	1
					Supported programmable keys			
User with supervision function	NT	N	N	N	N	N	Y	Y
Group	NT	N	N	Y	N	N	Y	Y
Queue	NT	N	N	Y	N	N	Y	Y
Conference Room with supervision function	NT	N	N	N	N	N	Y	Y
General Functions								
Online call recording	NT	N	N	N	N	N	N	N
Phone status	NT	N	N	Y	N	N	Y	Y
Sound recording	NT	N	N	Y	N	N	Y	Y
Call recording	NT	N	N	N	N	N	Y	Y
Incoming call filtering	NT	N	N	N	N	N	Y	Y
Do not disturb	NT	N	N	SK	N	N	SK	SK
Group interception	NT	N	N	N	N	N	N	N
Listen to online calls	NT	N	N	Y	N	N	Y	Y
Directory access	NT	N	N	Y	N	N	Y	Y
Filtering Boss - Secretary	NT	N	N	N	N	N	Y	Y
Transfers Functions								
Blind transfer	NT	N	N	N	N	N	N	N
Indirect transfer	NT	N	N	SK	N	N	SK	SK
Forwards Functions								

Table 1.2 – continued from previous page

	7905G	7906G	7911G	7912G	7920	7921G	7940G	7941
Disable all forwarding	NT	N	N	Y	N	N	Y	Y
Enable/Disable forwarding on no answer	NT	N	N	Y	N	N	Y	Y
Enable/Disable forwarding on busy	NT	N	N	Y	N	N	Y	Y
Enable/Disable forwarding unconditional	NT	N	N	Y	N	N	Y	Y
Voicemail Functions								
Enable voicemail with supervision function	NT	N	N	N	N	N	N	N
Activate voicemail	NT	N	N	N	N	N	Y	Y
Reach the voicemail	NT	N	N	SK	N	N	HK	HK
Delete messages from voicemail	NT	N	N	Y	N	N	Y	Y
Agent Functions								
Connect/Disconnect a static agent	NT	N	N	Y	N	N	Y	Y
Connect a static agent	NT	N	N	Y	N	N	Y	Y
Disconnect a static agent	NT	N	N	Y	N	N	Y	Y
Parking Functions								
Parking	NT	N	N	N	N	N	N	N
Parking position	NT	N	N	N	N	N	N	N
Paging Functions								
Paging	NT	N	N	Y	N	N	Y	Y

To install firmware for xivo-cisco-sccp plugins, you need to manually download the firmware files from the Cisco website and save them in the `/var/lib/xivo-provd/plugins/$plugin-name/var/cache` directory.

Note: The directory is created by XiVO when you install the plugin (i.e. xivo-cisco-sccp-legacy). If you create the directory manually, the installation may fail!

For example, if you have installed the `xivo-cisco-sccp-legacy` plugin and you want to install the `7940-7960-fw`, `networklocale` and `userlocale_fr_FR` package, you must:

- Go to http://www.cisco.com
- Click on "Log In" in the top right corner of the page, and then log in
- Click on the "Support" menu
- Click on the "Downloads" tab, then on "Voice & Unified Communications"
- Select "IP Telephony", then "Unified Communications Endpoints", then the model of your phone (in this example, the 7940G)
- Click on "Skinny Client Control Protocol (SCCP) software"
- Choose the same version as the one shown in the plugin
- Download the file with an extension ending in ".zip", which is usually the last file in the list
- In the XiVO web interface, you'll then be able to click on the "install" button for the firmware

The procedure is similar for the network locale and the user locale package, but:

- Instead of clicking on "Skinny Client Control Protocol (SCCP) software", click on "Unified Communications Manager Endpoints Locale Installer"
- Click on "Linux"
- Choose the same version of the one shown in the plugin
- For the network locale, download the file named "po-locale-combined-network.cop.sgn"
- For the user locale, download the file named "po-locale-$locale-name.cop.sgn, for example "po-locale-fr_FR.cop.sgn" for the "fr_FR" locale

- Both files must be placed in `/var/lib/xivo-provd/plugins/$plugin-name/var/cache` directory. Then install them in the XiVO Web Interface.

Note: Currently user and network locale 9.0.2 should be used for plugins xivo-sccp-legacy and xivo-cisco-sccp-9.0.3

Digium

Digium phones:

	D40	D50	D70
Provisioning	Y	NYT	Y
H-A	Y	NYT	Y
Directory XIVO	N	NYT	N
Funckeys	2	14	106
Supported programmable keys			
User with supervision function	N	NYT	N
Group	Y	NYT	Y
Queue	Y	NYT	Y
Conference Room with supervision function	Y	NYT	Y
General Functions			
Online call recording	N	NYT	N
Phone status	Y	NYT	Y
Sound recording	Y	NYT	Y
Call recording	Y	NYT	Y
Incoming call filtering	Y	NYT	Y
Do not disturb	HK	NYT	HK
Group interception	Y	NYT	Y
Listen to online calls	N	NYT	N
Directory access	N	NYT	N
Filtering Boss - Secretary	Y	NYT	Y
Transfers Functions			
Blind transfer	HK	NYT	HK
Indirect transfer	HK	NYT	HK
Forwards Functions			
Disable all forwarding	Y	NYT	Y
Enable/Disable forwarding on no answer	Y	NYT	Y
Enable/Disable forwarding on busy	Y	NYT	Y
Enable/Disable forwarding unconditional	Y	NYT	Y
Voicemail Functions			
Enable voicemail with supervision function	Y	NYT	Y
Activate voicemail	Y	NYT	Y
Reach the voicemail	HK	NYT	HK
Delete messages from voicemail	Y	NYT	Y
Agent Functions			
Connect/Disconnect a static agent	Y	NYT	Y
Connect a static agent	Y	NYT	Y
Disconnect a static agent	Y	NYT	Y
Parking Functions			
Parking	N	NYT	N
Parking position	N	NYT	N
Paging Functions			
Paging	Y	NYT	Y

Note: Some function keys are shared with line keys

Particularities:

- For best results, activate *DHCP Integration* on your XiVO.
- Impossible to do directed pickup using a BLF function key.
- Only supports DTMF in RFC2833 mode.
- Does not work reliably with Cisco ESW520 PoE switch. When connected to such a switch, the D40 tends to reboot randomly, and the D70 does not boot at all.
- It's important to not edit the phone configuration via the phones' web interface when using these phones with XiVO.
- Paging doesn't work.

Polycom

SoundPoint IP:

	|SoundPoint IP					|SoundStation IP	
	SPIP331	SPIP335	SPIP450	SPIP550	SPIP560	SPIP650	SPIP5000
Provisioning [4]	N [4]	Y	Y	Y	N [4]	N [4]	N [4]
H-A [3]	N	Y	N	Y	N	N	N
Directory XIVO	N	N	N	FK	N	N	N
Funckeys [2]	N	0	2	3	3	47	0
Supported programmable keys							
User with supervision function	NYT	N	NYT	Y	NYT	NYT	NYT
Group	NYT	N	NYT	Y	NYT	NYT	NYT
Queue	NYT	N	NYT	Y	NYT	NYT	NYT
Conference Room with supervision function	NYT	N	NYT	Y	NYT	NYT	NYT
General Functions							
Online call recording	NYT	N	NYT	N	NYT	NYT	NYT
Phone status	NYT	N	NYT	Y	NYT	NYT	NYT
Sound recording	NYT	N	NYT	Y	NYT	NYT	NYT
Call recording	NYT	N	NYT	Y	NYT	NYT	NYT
Incoming call filtering	NYT	N	NYT	Y	NYT	NYT	NYT
Do not disturb	NYT	SK	NYT	HK	NYT	NYT	NYT
Group interception	NYT	N	NYT	Y	NYT	NYT	NYT
Listen to online calls	NYT	N	NYT	Y	NYT	NYT	NYT
Directory access	NYT	N	NYT	Y	NYT	NYT	NYT
Filtering Boss - Secretary	NYT	N	NYT	Y	NYT	NYT	NYT
Transfers Functions							
Blind transfer	NYT	SK	NYT	N	NYT	NYT	NYT
Indirect transfer	NYT	SK	NYT	HK	NYT	NYT	NYT
Forwards Functions							
Disable all forwarding	NYT	N	NYT	Y	NYT	NYT	NYT
Enable/Disable forwarding on no answer	NYT	SK	NYT	Y	NYT	NYT	NYT
Enable/Disable forwarding on busy	NYT	SK	NYT	Y	NYT	NYT	NYT
Enable/Disable forwarding unconditional	NYT	SK	NYT	Y	NYT	NYT	NYT
Voicemail Functions							
Enable voicemail with supervision function	NYT	N	NYT	Y	NYT	NYT	NYT
Activate voicemail	NYT	N	NYT	Y	NYT	NYT	NYT
Reach the voicemail	NYT	SK	NYT	HK	NYT	NYT	NYT
Delete messages from voicemail	NYT	N	NYT	Y	NYT	NYT	NYT
Agent Functions							
Connect/Disconnect a static agent	NYT	N	NYT	Y	NYT	NYT	NYT
Connect a static agent	NYT	N	NYT	Y	NYT	NYT	NYT
Disconnect a static agent	NYT	N	NYT	Y	NYT	NYT	NYT

Table 1.4 – continued from previous page

	|SoundPoint IP					|SoundStation IP	
Parking Functions							
Parking	NYT	N	NYT	N	NYT	NYT	NYT
Parking position	NYT	N	NYT	N	NYT	NYT	NYT
Paging Functions							
Paging	NYT	N	NYT	Y	NYT	NYT	NYT

Particularities:

- Directed pickup doesn't work when using a BLF function key. The workaround is to put both the user and the supervised user in the same call pickup group.
- VVX: the french translation is incomplete.

Note: (XiVO HA cluster) BLF function key saved on the master node are not available.

Polycom® SoundPoint® IP Backlit Expansion Module are supported.

Snom

	370	710	715	720	760	821	870
Provisioning [1]	Y	Y	Y	Y	Y	Y	Y
H-A [3]	Y	Y	Y	Y	Y	Y	Y
Directory XIVO	HK	SK	SK	HK	HK	HK	HK
Funckeys [2]	12	5	5	18	16	12	15
	Supported programmable keys						
User with supervision function	Y	Y	Y	Y	Y	Y	Y
Group	Y	Y	Y	Y	Y	Y	Y
Queue	Y	Y	Y	Y	Y	Y	Y
Conference Room with supervision function	Y	Y	Y	Y	Y	Y	Y
General Functions							
Online call recording	N	N	N	N	N	N	N
Phone status	Y	Y	Y	Y	Y	Y	Y
Sound recording	Y	Y	Y	Y	Y	Y	Y
Call recording	Y	Y	Y	Y	Y	Y	Y
Incoming call filtering	Y	Y	Y	Y	Y	Y	Y
Do not disturb	HK	SK	SK	HK	HK	HK	HK
Group interception	Y	Y	Y	Y	Y	Y	Y
Listen to online calls	Y	Y	Y	Y	Y	Y	Y
Directory access	Y	Y	Y	Y	Y	Y	Y
Filtering Boss - Secretary	Y	Y	Y	Y	Y	Y	Y
Transfers Functions							
Blind transfer	Y	SK	SK	HK	HK	HK	HK
Indirect transfer	Y	SK	SK	HK	HK	HK	HK
Forwards Functions							
Disable all forwarding	Y	Y	Y	Y	Y	Y	Y
Enable/Disable forwarding on no answer	Y	Y	Y	Y	Y	Y	Y
Enable/Disable forwarding on busy	Y	Y	Y	Y	Y	Y	Y
Enable/Disable forwarding unconditional	Y	Y	Y	Y	Y	Y	Y
Voicemail Functions							
Enable voicemail with supervision function	Y	Y	Y	Y	Y	Y	Y
Activate voicemail	Y	Y	Y	Y	Y	Y	Y
Reach the voicemail	HK	HK	HK	HK	HK	HK	HK
Delete messages from voicemail	Y	Y	Y	Y	Y	Y	Y
Agent Functions							

Continued on next page

Table 1.5 – continued from previous page

	370	710	715	720	760	821	870
Connect/Disconnect a static agent	Y	Y	Y	Y	Y	Y	Y
Connect a static agent	Y	Y	Y	Y	Y	Y	Y
Disconnect a static agent	Y	Y	Y	Y	Y	Y	Y
Parking Functions							
Parking	Y	N	N	N	N	Y	Y
Parking position	Y	N	N	N	N	Y	Y
Paging Functions							
Paging	Y	Y	Y	Y	Y	Y	Y

Snom Vision – the expansion module for snom 8xx series VoIP telephones are supported.

Snom extension modules V2.0 are supported.

Note: For some models, function keys are shared with line keys

Warning: If you are using Snom phones with HA, you should not assign multiple lines to the same device.

There's a known issue with the provisioning of Snom phones in XiVO:

- After a factory reset of a phone, if no language and timezone are set for the "default config device" in *XiVO → Configuration → Provisioning → Template device*, you will be forced to select a default language and timezone on the phone UI.

Yealink

	T19P	T20P	T21P	T22P	T26P	T28P	T32G	T38G	T4
Provisioning [1]	Y	Y	Y	Y	Y	NT	NT [4]	Y	Y
H-A [3]	Y	Y	Y	Y	Y	Y	N	N	Y
Directory XIVO	MN	N	MN	MN	MN	SK	NT	SK	MN
Funckeys [2]	N	2	2	3	13	16	3	16	15
	Supported programmable keys								
User with supervision function	N	Y	Y	Y	Y	Y	NYT	Y	Y
Group	N	Y	Y	Y	Y	Y	NYT	Y	Y
Queue	N	Y	Y	Y	Y	Y	NYT	Y	Y
Conference Room with supervision function	N	Y	Y	Y	Y	Y	NYT	Y	Y
General Functions									
Online call recording	N	N	N	N	N	N	NYT	N	N
Phone status	N	Y	Y	Y	Y	Y	NYT	Y	Y
Sound recording	N	Y	Y	Y	Y	Y	NYT	Y	Y
Call recording	N	Y	Y	Y	Y	Y	NYT	Y	Y
Incoming call filtering	N	Y	Y	Y	Y	Y	NYT	Y	Y
Do not disturb	N	Y	SK	SK	SK	SK	NYT	SK	SK
Group interception	N	Y	Y	Y	Y	Y	NYT	Y	Y
Listen to online calls	N	Y	Y	Y	Y	Y	NYT	Y	Y
Directory access	N	Y	Y	Y	Y	Y	NYT	Y	Y
Filtering Boss - Secretary	N	Y	Y	Y	Y	Y	NYT	Y	Y
Transfers Functions									
Blind transfer	N	HK	HK	HK	HK	HK	NYT	HK	SK
Indirect transfer	N	HK	HK	HK	HK	HK	NYT	HK	SK
Forwards Functions									
Disable all forwarding	N	Y	Y	Y	Y	Y	NYT	Y	Y
Enable/Disable forwarding on no answer	N	Y	Y	Y	Y	Y	NYT	Y	Y
Enable/Disable forwarding on busy	N	Y	Y	Y	Y	Y	NYT	Y	Y

Table 1.6 – continued from previous page

	T19P	T20P	T21P	T22P	T26P	T28P	T32G	T38G	T4
Enable/Disable forwarding unconditional	N	Y	Y	Y	Y	Y	NYT	Y	Y
Voicemail Functions									
Enable voicemail with supervision function	N	Y	Y	Y	Y	Y	NYT	Y	Y
Activate voicemail	N	Y	Y	Y	Y	Y	NYT	Y	Y
Reach the voicemail	N	HK	HK	HK	HK	HK	NYT	HK	HK
Delete messages from voicemail	N	Y	Y	Y	Y	Y	NYT	Y	Y
Agent Functions									
Connect/Disconnect a static agent	N	Y	Y	Y	Y	Y	NYT	Y	Y
Connect a static agent	N	Y	Y	Y	Y	Y	NYT	Y	Y
Disconnect a static agent	N	Y	Y	Y	Y	Y	NYT	Y	Y
Parking Functions									
Parking	N	Y	Y	Y	Y	Y	NYT	N	Y
Parking position	N	Y	Y	Y	Y	Y	NYT	N	Y
Paging Functions									
Paging	N	Y	Y	Y	Y	Y	NYT	NYT	Y

Note: Some function keys are shared with line keys

The EXP38 ,EXP39 and EXP40 expansion modules are supported.

Compatible device (Community support)

The following phones are only supported by the community. In other words, maintenance, bug corrections and features are developed by members of the XiVO community. XiVO does not officially endorse support for these phones.

`xivo-provd` plugins for these devices can be installed from the *"community supported devices" repository*.

Aastra

6700i and 9000i series:

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
6730i	No	8	Yes
6753i	Yes	6	Yes
6757i	Yes	30	Yes
9143i	Yes	7	Yes
9480i	No	6	Yes
9480CT	No	6	Yes

Alcatel-Lucent

IP Touch series:

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
4008 Extended Edition	Yes	4	No
4018 Extended Edition	Yes	4	No

Note that you *must not* download the firmware for these phones unless you agree to the fact it comes from a non-official source.

For the plugin to work fully, you need these additional packages:

```
apt-get install p7zip python-pexpect telnet
```

Avaya

1200 series IP Deskphones (previously known as Nortel IP Phones):

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
1220 IP	Yes	0	No
1230 IP	No	0	No

Cisco

Cisco Small Business SPA300 series:

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
SPA301	No	1	No
SPA303	No	3	No

Note: Function keys are shared with line keys for all SPA phones

Cisco Small Business SPA500 series:

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
SPA501G	Yes	8	No
SPA502G	No	1	No
SPA504G	Yes	4	No
SPA508G	Yes	8	No
SPA509G	No	12	No
SPA512G	No	1	No
SPA514G	No	4	No
SPA525G	Yes	5	No
SPA525G2	No	5	No

The SPA500 expansion module is supported.

Cisco Small Business IP Phones (previously known as Linksys IP Phones)

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
SPA901	No	1	No
SPA921	No	1	No
SPA922	No	1	No
SPA941	No	4	No
SPA942	Yes	4	No
SPA962	Yes	6	No

Note: You must install the firmware of each SPA9xx phones you are using since they reboot in loop when they can't find their firmware.

The SPA932 expansion module is supported.

ATAs:

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
PAP2	No	0	No
SPA2102	No	0	No
SPA8800	No	0	No

For best results, activate *DHCP Integration* on your XiVO.

Note: These devices can be used to connect Faxes. For better success with faxes some parameters must be changed. You can read the *Using analog gateways* section.

Note: If you want to manually resynchronize the configuration from the ATA device you should use the following url:

```
http://ATA_IP/admin/resync?http://XIVO_IP:8667/CONF_FILE
```

where :

- *ATA_IP* is the IP address of the ATA,
- *XIVO_IP* is the IP address of your XiVO,
- *CONF_FILE* is one of `spa2102.cfg`, `spa8000.cfg`

Gigaset

Also known as Siemens.

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
C470 IP	No	0	No
C475 IP	No	0	No
C590 IP	No	0	No
C595 IP	No	0	No
C610 IP	No	0	No
C610A IP	No	0	No
S675 IP	No	0	No
S685 IP	No	0	No
N300 IP	No	0	No
N300A IP	No	0	No
N510 IP PRO	No	0	No

Jitsi

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
Jitsi	Yes	—	No

Panasonic

Panasonic KX-HTXXX series:

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
KX-HT113	No	—	No
KX-HT123	No	—	No
KX-HT133	No	—	No
KX-HT136	No	—	No

Note: This phone is for testing for the moment

Polycom

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
SPIP320	No	0	No
SPIP321	No	0	No
SPIP330	No	0	No
SPIP430	No	0	No
SPIP501	Yes	0	No
SPIP600	No	0	No
SPIP601	No	0	No
SPIP670	No	47	No

SoundStation IP:

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
SPIP4000	No	0	No

Others:

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
VVX1500	No	0	No

Snom

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
300	No	6	Yes
320	Yes	12	Yes
360	No	—	Yes
820	Yes	4	Yes
MP	No	—	Yes
PA1	No	0	Yes

Note: For some models, function keys are shared with line keys

Warning: If you are using Snom phones with HA, you should not assign multiple lines to the same device.

There's a known issue with the provisioning of Snom phones in XiVO:

- After a factory reset of a phone, if no language and timezone are set for the "default config device" in *XiVO → Configuration → Provisioning → Template device*, you will be forced to select a default language and timezone on the phone UI.

Technicolor

Previously known as Thomson:

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
ST2022	No	—	No
ST2030	Yes	10	No

Note: Function keys are shared with line keys

Yealink

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
T20P	No	2	No
T26P	No	13	No

Note: Some function keys are shared with line keys

Zenitel

Model	Tested [1]	Fkeys [2]	XiVO HA [3]
IP station	Yes	1	No

Caption :

1.8 Administration

1.8.1 Advanced Configuration

This section describes the advanced system configuration.

XiVO General Settings

XiVO offers the possibility to configure the general settings via the *Configuration → Management → General* page.

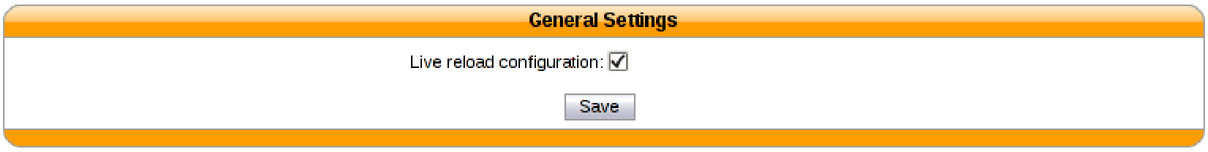


Figure 1.26: Configure XiVO General Settings

Live reload configuration permit to reload its configuration on command received from WEBI (this option is enabled by default).

Certificates

XiVO offers the possibility to create and manage X.509 certificates via the the *Configuration → Management → Certificates* page.

These certificates can be used for:

- enabling SIP TLS
- enabling encryption between the CTI server and the XiVO clients

Creating certificates

You can add a certificate by clicking on the add button at the top right of the page. You'll then be shown this page:

You should look at the examples if you don't know which attributes to set when creating your certificates.

Removing certificates

When removing a certificate, you should remove all the files related to that certificates.

> **Warning:** If you remove a certificate that is used somewhere in XiVO, then you need to manually reconfigure that portion of XiVO.
> For example, if you remove the certificate files used for SIP TLS, then you need to manually disable SIP TLS or asterisk will look for certificate file but it won't be able to find them.

Examples

In the following examples, if a field is not specified than you should leave it at its default value.

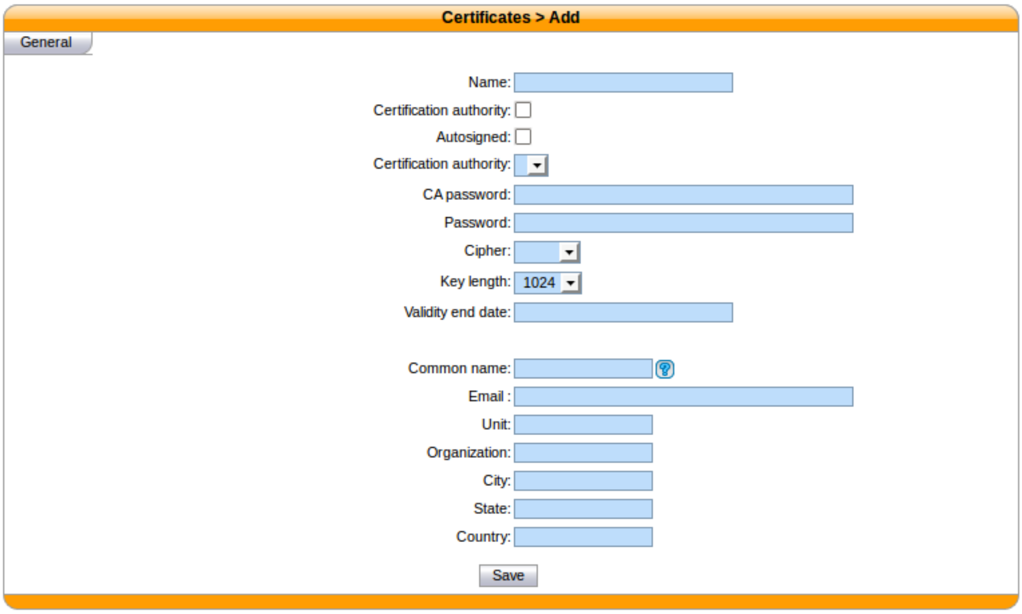


Figure 1.27: Adding a certificate

Creating certificates for SIP TLS You need to create both a CA certificate and a server certificate.

CA certificate:

- *Name* : phones-CA
- *Certification authority (checkbox)* : checked
- *Autosigned* : checked
- *Valid end date* : at least one month in the future
- *Common name* : the FQDN (Fully Qualified Domain Name) of your XiVO
- *Organization* : your organization's name, or blank
- *Email* : your email or organization's email

Server certificate:

- *Name* : phones
- *Certification authority (select)* : phones-CA
- *Valid end date* : at least one month in the future
- *Common name* : the FQDN of your XiVO
- *Organization* : your organization's name, or blank
- *Email* : your email or organization's email

Creating certificate for CTI server

- *Name* : xivo-ctid
- *Autosigned* : checked
- *Valid end date* : at least one month in the future
- *Common name* : the FQDN of your XiVO
- *Organization* : your organization's name, or blank
- *Email* : your email or organization's email

Warning: You must *not* set a password for the certificate. If the certificate is password protected, the CTI server will not be able to use it.

LDAP

XiVO offers the possibility to integrate LDAP servers. Once configured properly, you'll be able to search your LDAP servers directly from your phones (if they support this feature).

Add a LDAP Server

You can add a LDAP server by clicking on the add button at the top right corner of the *Configuration* → *Management* → *LDAP Servers* page. You'll then be shown this page:

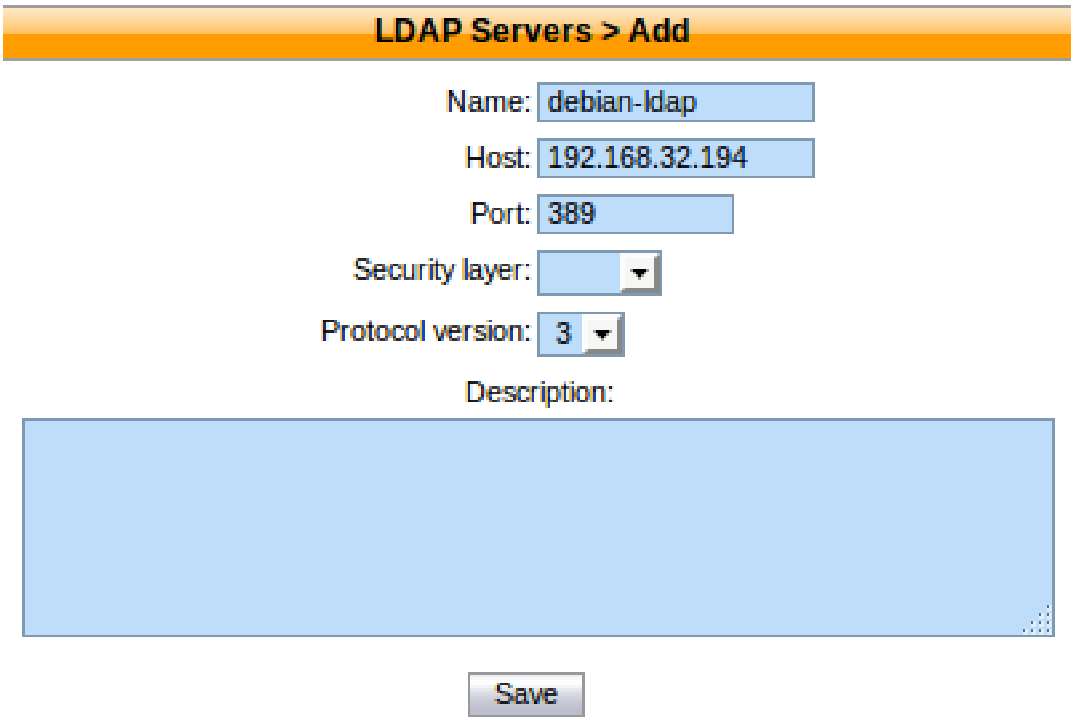


Figure 1.28: Adding a LDAP server

Enter the following information:

- Name: the server's display name
- Host: the hostname or IP address
- Port: the port number (default: 389)
- Security layer: select SSL if it is activated on your server and you want to use it (default: disabled)
  - SSL means TLS/SSL (doesn't mean StartTLS) and port 636 should then be used
- Protocol version: the LDAP protocol version (default: 3)

Warning: When editing an LDAP server, you'll have to restart the CTI server for the changes to be taken into account.

Notes on SSL/TLS usage If you are using SSL with an LDAP server that is using a CA certificate from an unknown certificate authority, you'll have to put the certificate file as a single file ending with `.crt` into `/usr/local/share/ca-certificates` and run `update-ca-certificates`.

You also need to make sure that the `/etc/ldap/ldap.conf` file contains a line `TLS_CACERT /etc/ssl/certs/ca-certificates.crt`.

After that, restart spawn-fcgi with `/etc/init.d/spawn-fcgi restart`.

Also, make sure to use the FQDN of the server in the host field when using SSL. The host field must match exactly what's in the CN attribute of the server certificate.

Add a LDAP Filter

Next thing to do after adding a LDAP server is to create a LDAP filter via the *Services* → *IPBX configuration* → *LDAP Filters* page.

LDAP filters define the information that will be searched and displayed when you do a directory search.

You can add a LDAP filter by clicking on the add button at the top right of the page. You'll then be shown this page:

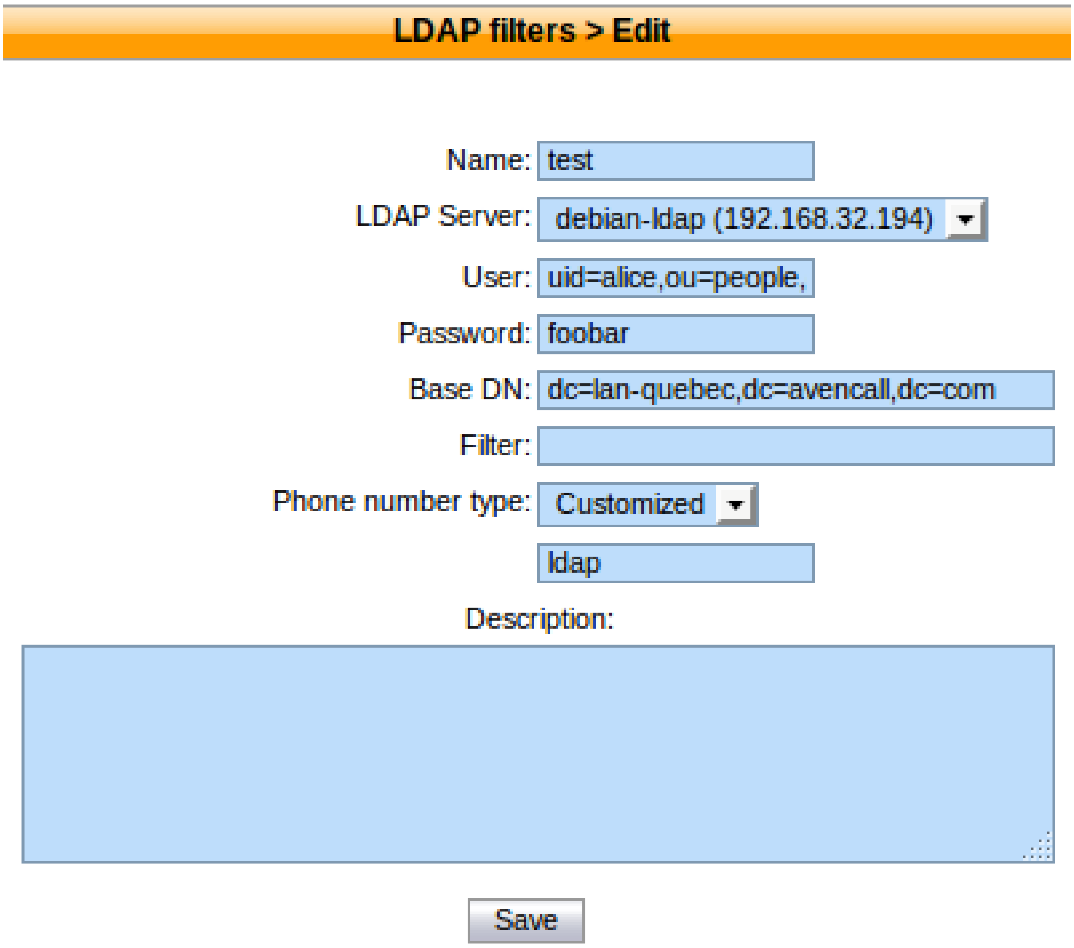


Figure 1.29: Adding a LDAP Filter

Enter the following information:

- Name: the filter's display name
- LDAP server: the LDAP server this filter applies to

- User: the `dn` of the user used to do search requests
- Password: the password of the given user
- Base DN: the base `dn` of search requests
- Filter: if specified, *it replace the default filter*
- Phone number type: this string is appended next to each result display name

You'll also probably need to modify some values in the *Attributes* tab:

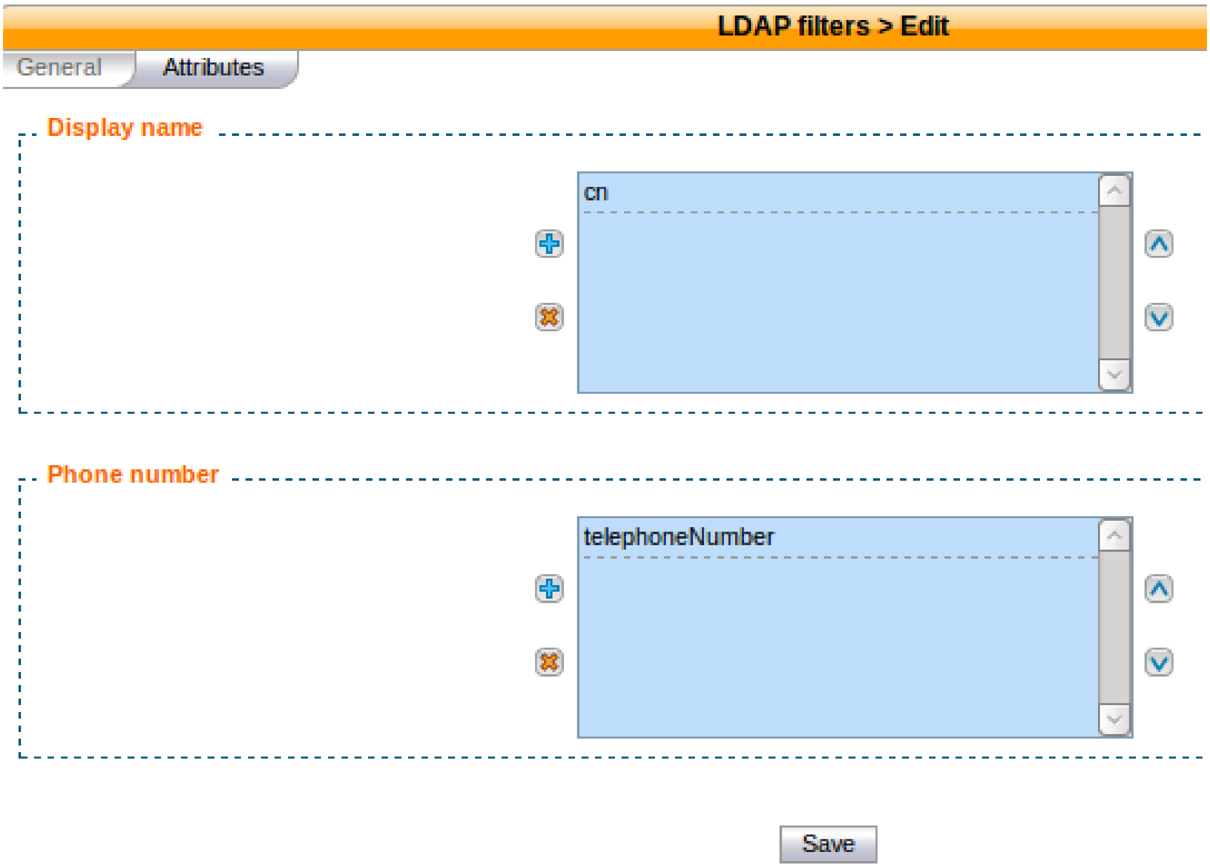


Figure 1.30: Adding a LDAP Filter

In the *Display name* section, add and order the attributes that are going be used to display the results. The first attribute will be used for each result which have this attribute, else the second will be used, etc.

The *Phone number* section is similar, but is used for the phone number in the results.

Use a Custom Filter In some cases, you might have to use a custom filter for your search requests instead of the default filter.

By default, the search tries to match any attribute you choose in the *Attributes* tab.

In custom filters, occurrence of the pattern `%Q` is replaced by what the user entered on its phone.

Here's some examples of custom filters:

- `cn=*%Q*`
- `&(cn=*%Q*)(mail=*@example.org)`
- `|(cn=*%Q*)(displayName=*%Q*)`

Add a LDAP filter to the Phonebook

In the *Services* → *General settings* → *Phonebook* page, click on the *LDAP filters* page and add your filter to the list of enabled filters.

Please refer to the *Remote directory* section in order to properly configure a remote directory.

You'll then be able to search your LDAP server directly from your phone and dial from the displayed results.

Use with CTI Server and Client XiVO

Add a LDAP Directory Filter to the CTI Server In the *Services* → *CTI Server* → *Directories* → *Definitions* page, click on the add button

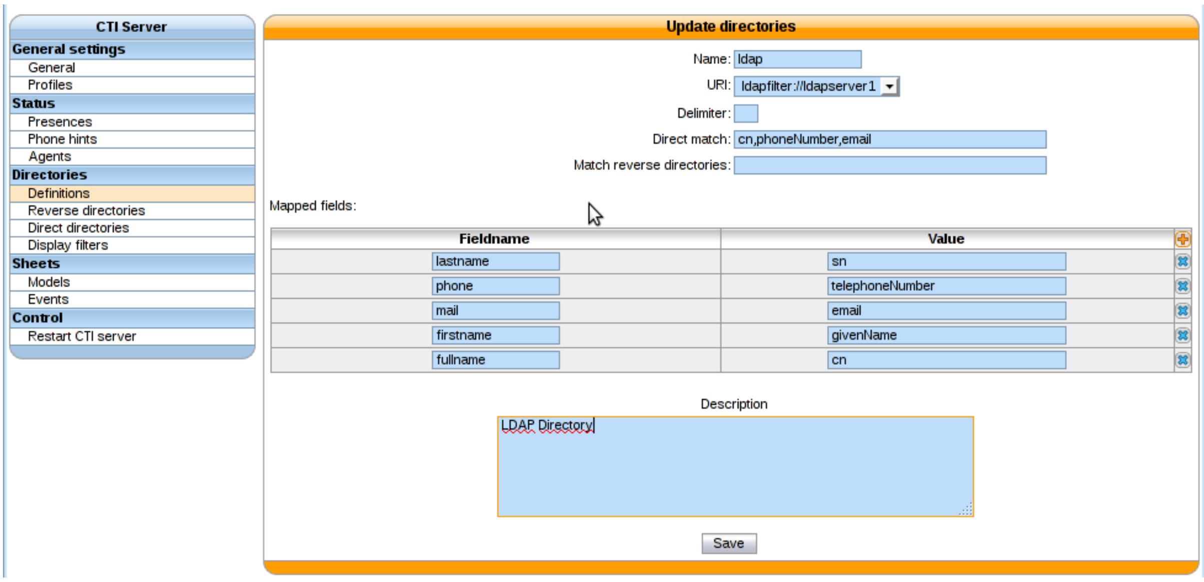


Figure 1.31: Adding a directory LDAP Filter to CTI Server

- `Direct match` use to search into this field
- `Match reverse directory` use to search into this field for the reverse directory
- `Fieldname/value` match to the CTI field> server> | field> LDAP> server.

Add a LDAP Directory to the CTI Server In the *Services* → *CTI Server* → *Directories* → *Direct directories* page, click on the edit button for `default` directory

To use this directory, you must then add to the list of searchable directories.

> **Warning:** The CTI server settings resonates in contexts. This means creating a context for each CTI context of membership of your users who will examine the "Directories".

- eg: CTI Context `default` for users in context `default`

Restart CTI Server

1.8.2 Boss Secretary Filter

The boss secretary filter allow to set a secretary or a boss role to a user. Filters can then be created to filter calls directed to a boss using different strategies.

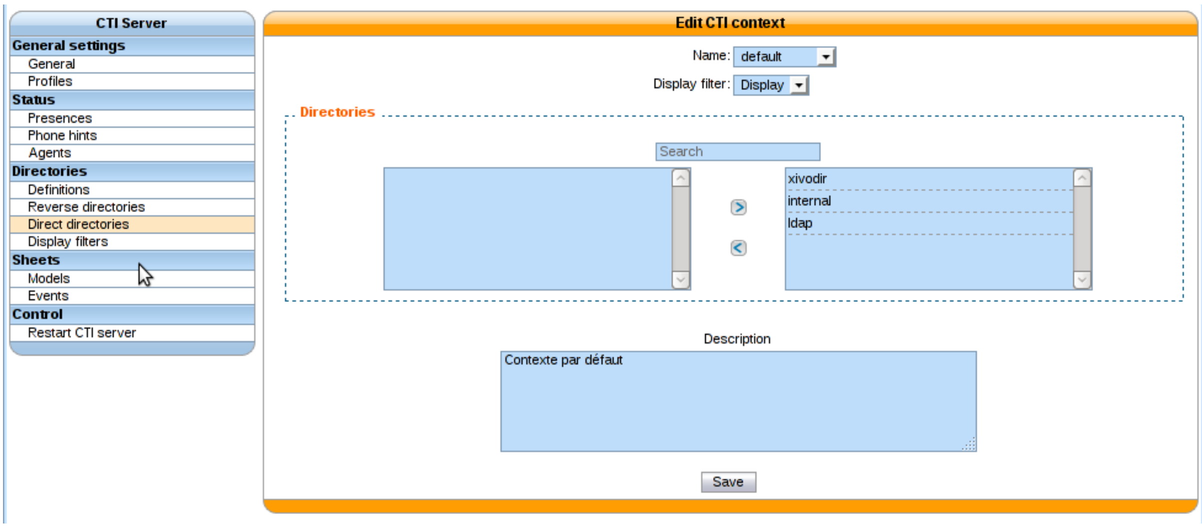


Figure 1.32: Adding a directory LDAP to CTI Server

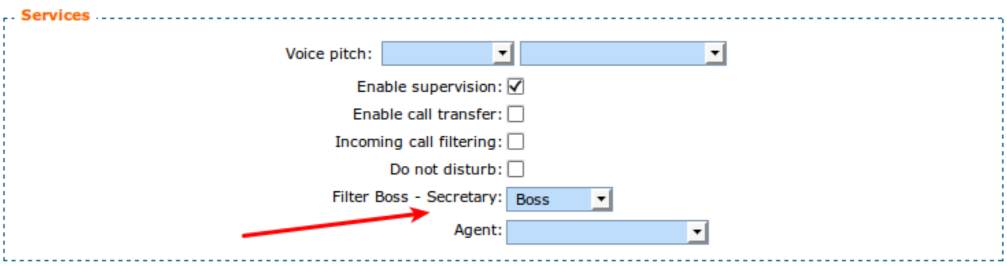
Quick Summary

In order to be able to use the boss secretary filter you have to :

- Select a boss role for one the users
- Select a secretary role for one ot the users
- Create a filter to set a strategy for this boss secretary filter
- Add a function key for the user boss and the user secretary

Defining a Role

The secretary or boss role can be set in the user's configuration page under the service tab. To use this feature, at least one boss and one secretary must be defined.

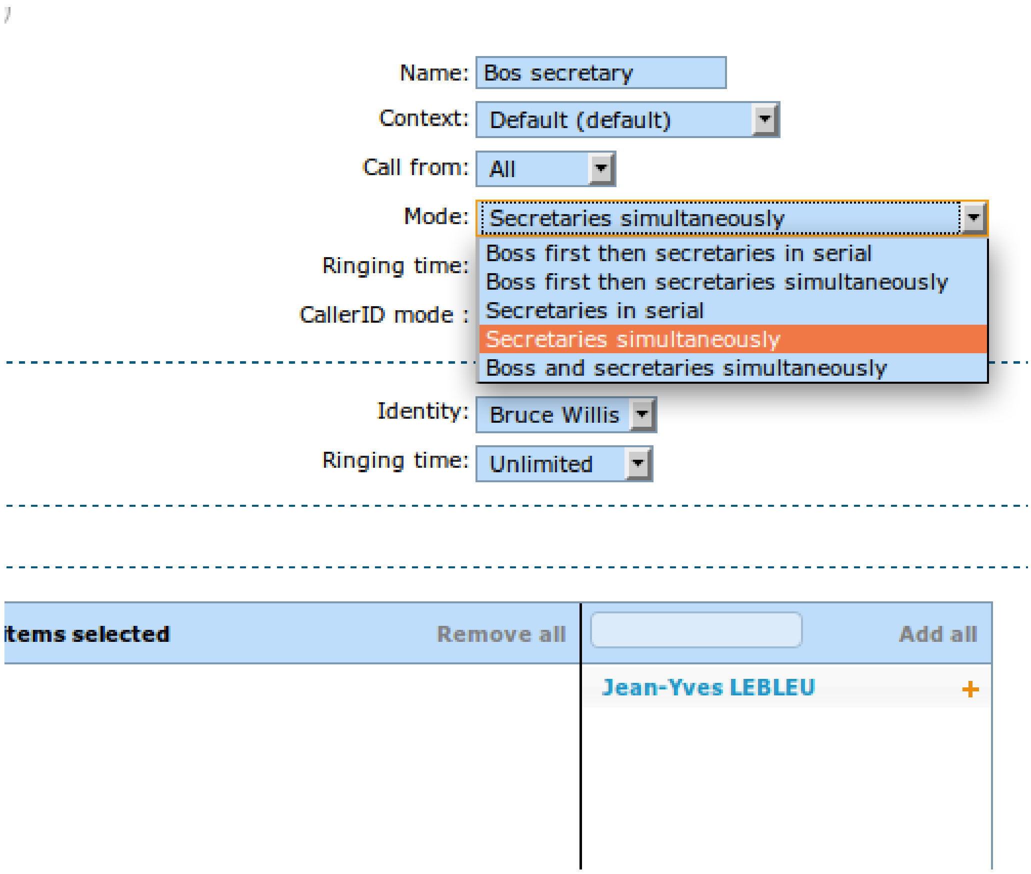


Creating a Filter

The filter is used to associate a boss to one or many secretaries and to set a ring strategy. The call filter is added in the *Services* → *IPBX* → *Call management* → *Call filters* page.

Different ringing strategies can be applied :

- Boss rings first then all secretaries one by one
- Boss rings first then secretaries are all ringing simultaneously
- Secretaries ring one by one
- Secretaries are all ringing simultaneously



- Boss and secretaries are ringing simultaneously
- Change the caller id if the secretary wants to know which boss was initialy called

When one of serial strategies is used, the first secretary called is the last in the list. The order can be modified by drag and drop in the list.

Usage

The call filter function can be activated and deactivated by the boss or the secretary using the *37 extension. The extension is defined in *IPBX services > Extensions*.

The call filter has to be activated for each secretary if more than one is defined for a given boss.

The extension to use is `*37<callfilter member id>`.

In this example, you would set 2 `Func Keys` `*373` and `*374` on the Boss.

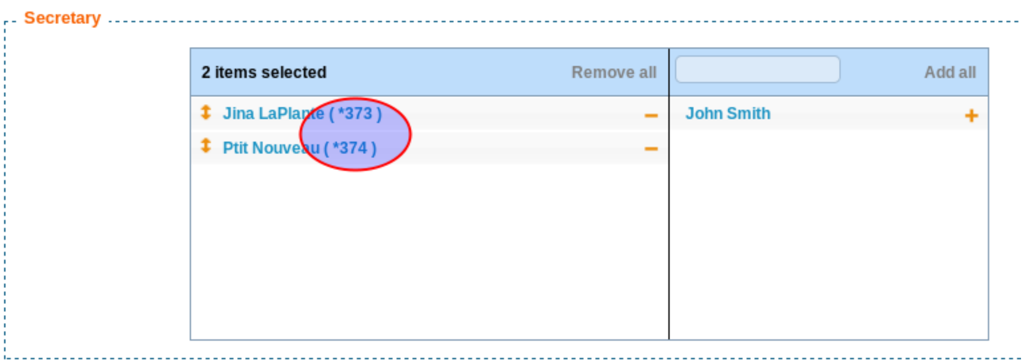
On the secretary `Jina LaPlante` you would set `*373`.

On the secretary `Ptit Nouveau` you would set `*374`.

Function Keys

A more convenient way to active the boss secretary filter is to assign a function key on the boss' phone or the secretary's phone. In the user's configuration under `Func Keys`. A function key can be added for each secretaries of a boss.

If supervision is activated, the key will light up when filter is activated for this secretary. If a secretary also has a function key on the same boss/secretary combination the function key's BLF will be in sync between each phones.



Warning: With SCCP phones, you must configure a custom `Func Keys`.

1.8.3 Call Permissions

You can manage call permissions via the *Services* → *IPBX* → *Call management* → *Call permissions* page.

Call permissions can be used for:

- denying a user from calling a specific extension
- denying a user of a group from calling a specific extension
- denying a specific extension on a specific outgoing call from being called
- denying an incoming call coming from a specific extension from calling you

More than one extension can match a given call permission, either by specifying more than one extension for that permission or by using extension patterns.

You can also create permissions that allow a specific extension to be called instead of being denied. This make it possible to create a general "deny all" permission and then an "allow for some" one.

Finally, instead of unconditionally denying calling a specific extension, call permissions can instead challenge the user for a password to be able to call that extension.

As you can see, you can do a lot of things with XiVO's call permissions. They can be used to create fairly complex rules. That said, it is probably *not* a good idea to so because it's pretty sure you'll get it somehow wrong.

Examples

Note that when creating or editing a call permission, you must at least:

- fill the *Name* field
- have one extension / extension pattern in the *Extensions* field

Denying a user from calling a specific extension

- Add the extension in the extensions list
- In the *Users* tab, select the user

Warning: The extension can be anything but it will only work if it's the extension of a user or an extension that pass through an outgoing call. It does *not* work, for example, if the extension is the number of a conference room.

Denying a user of a group from calling a specific extension

First, you must create a group and add the user to this group. Note that groups aren't required to have a number.

Then,

- Add the extension in the extensions list
- In the *Groups* tab, select the group

Denying users from calling a specific extension on a specific outgoing call

- Add the extension in the extensions list
- In the *Outgoing calls* tab, select the outgoing call

Note that selecting both a user and an outgoing call for the same call permission doesn't mean the call permission applies only to that user. In fact, it means that the user can't call that extension and that the extension can't be called on the specific outgoing call. This in redundant and you will get the same result by not selecting the user.

Denying an incoming call coming from a specific extension from calling you

Call permissions on incoming calls are semantically different from the other scenarios since the extension that you add to the permission will match the extension of the caller (i.e. the caller number) and *not* the extension that the caller dialed (i.e. the callee number).

- Add the extension in the extensions list.
- In the *Incoming calls* tab, select the incoming call

1.8.4 Call Logs

Call logs are pre-generated from CEL entries. The generation is done automatically by xivo-call-logd. xivo-call-logs is also run nightly to generate call logs from CEL that were missed by xivo-call-logd.

Search Dashboard

Call logs can be accessed using the menu *Services* → *IPBX* → *Call management* → *Call Logs* page.

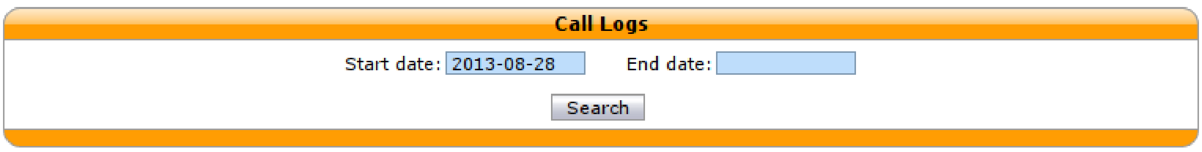


Figure 1.33: Calls Records Dashboard

Call logs are presented in a CSV file. The CSV specifications are detailed in *the REST API documentation*.

Specifying no start date returns all available call logs. Specifying a start date and no end date returns all call logs from start date until now.

REST API

Call logs are also available from the REST API. See *Call Logs*.

Manual generation

Call logs can also be generated manually. To do so, log on to the target XiVO server and run:

```
xivo-call-logs
```

To avoid running for too long in one time, the call logs generation is limited to the N last unprocessed CEL entries (default 20,000). This means that successive calls to `xivo-call-logs` will process N more CELs, making about N/10 more calls available in call logs, going further back in history, while processing new calls as well.

You can specify the number of CEL entries to consider. For example, to generate calls using the 100,000 last unprocessed CEL entries:

```
xivo-call-logs -c 100000
```

1.8.5 Conference Room

Adding a conference room

In this example, we'll add a conference room with number 1010.

First, you need to define a conference room number range for the ''default'' context via the ''Services / IPBX / IPBX configuration / Contexts'' page.

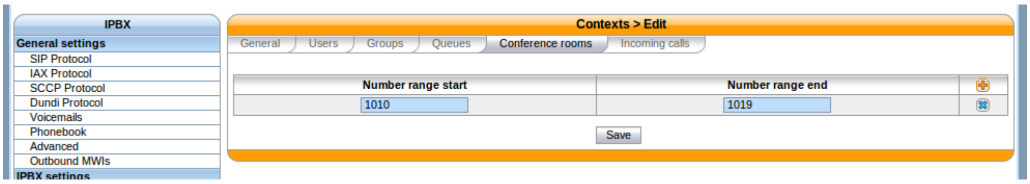


Figure 1.34: Adding a conference room number range to the default context

You can then create a conference room via the ''Services / IPBX / IPBX settings / Conference rooms'' page.

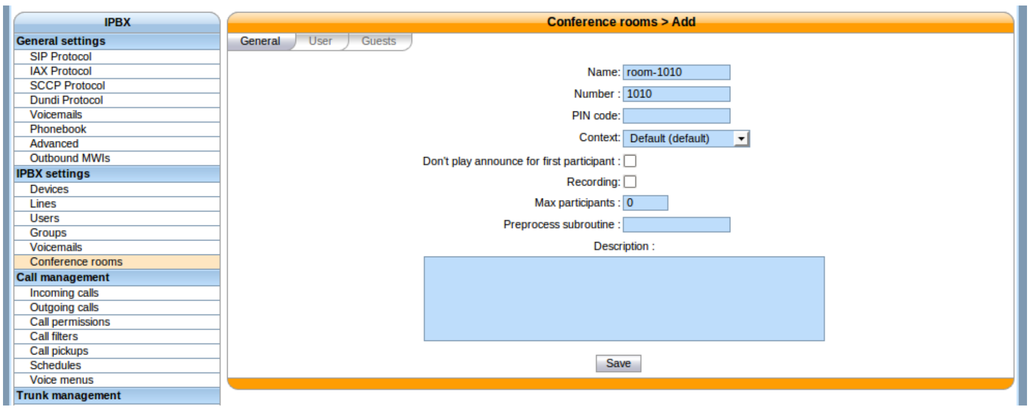


Figure 1.35: Creating conference room 1010

In this example, we have only filled the ''Name'' and ''Number'' fields, the others have been left to their default value.

As you can see, there's quite a few options when adding / editing a conference room. Here's a description of the most common one:

General / PIN code Protects your conference room with a PIN number. People trying to join the room will be asked for the PIN code.

General / Don't play announce for first participant Don't play the "you are currently the only person in this conference" for the first participant.

General / Max participants Limits the number of participants in the conference room. A value of 0 means unlimited.

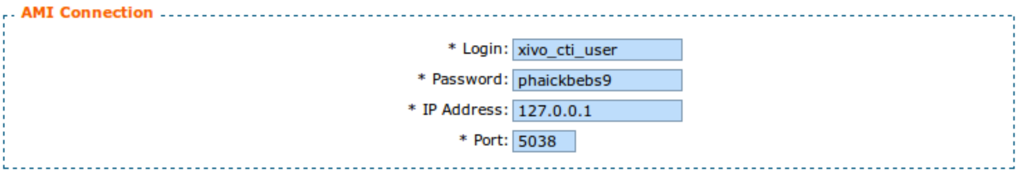
1.8.6 CTI Server

The CTI server configuration options can be found in the web-interface under the services tab.

General Options

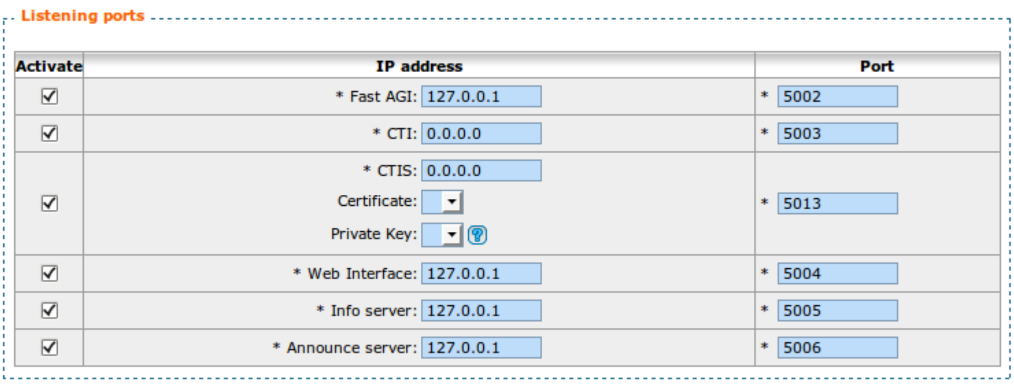
The general options allow the administrator to manage network connections between the CTI server and other services and clients.

The section named AMI connection allows the administrator to configure the information required to connect to the Asterisk Manager Interface (AMI). These fields should match the entries in `/etc/asterisk/manager.conf`.



The section named `Listening Ports` allows the administrator to specify listening addresses and ports for the CTI server's interfaces.

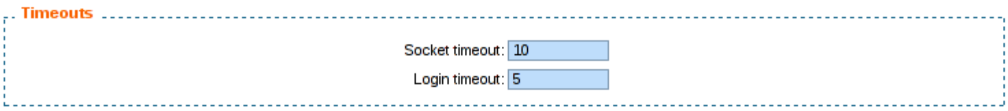
- Fast AGI is the CTI server's entry point for the Asterisk dialplan. This address and port have nothing to do with the listening port and address of xivo-agid.
- CTI and CTIs are for the client's connection and secure connection respectively.
- Web Interface is for the port used to receive events from the XiVO web interface
- Info server a debugging console to do some introspection on the state of the CTI server
- Announce is used to notify the CTI server when a dialplan reload is requested



The timeout section allow the administrator to configure multiple timeouts.

- Socket timeout is the default timeout used for network connections.
- Login timeout is the timeout before a CTI connection is dropped if the authentication is not completed.

Parting options are used to isolate XiVO users from each other. These options should be used when using the same XiVO for different enterprises.

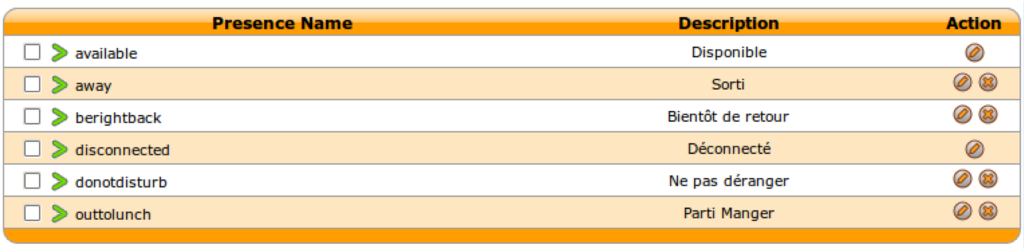


Context separation is based on the user's line context. A user with no line is not the member of any context and will not be able to do anything with the CTI client.

Note: The CTI Server must be restarted to take into account this parameter.

Presence Option

In the *Status* menu, under *Presences*, you can edit presences group. The default presence group is xivo. When editing a group, you will see a list of presences and there descriptions.



Presence Name	Description	Action
available	Disponible	
away	Sorti	
berightback	Bientôt de retour	
disconnected	Déconnecté	
donotdisturb	Ne pas déranger	
outtolunch	Parti Manger	

Available configuration

- *Presence name* is the name of the presence
- *Display name* is the human readable representation of this presence
- *Color status* is the color associated to this presence
- *Other reachable statuses* is the list of presence that can be switched from this presence state
- *Actions* are post selection actions that are triggered by selecting this presence

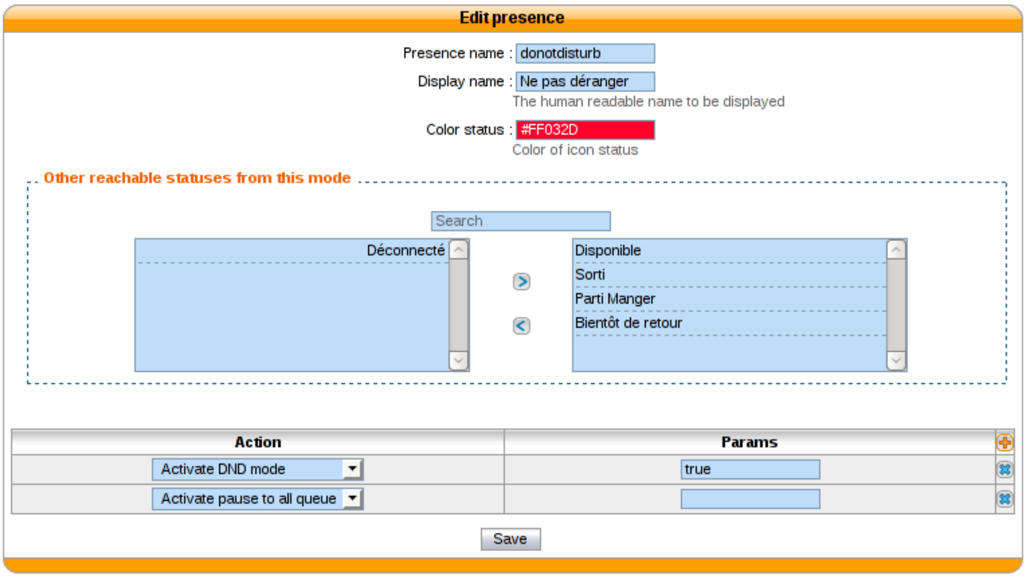
Actions

action	param
Enable DND	*{'true','false'}*
Pause agent in all queues	
Unpause agent in all queues	
Agent logoff	

Enable encryption

To enable encryption of CTI communications between server and clients, you have to create a certificate in *Configuration → Certificates*.

Then, go in the menu *CTI Server → General settings → General*, and in the section `Listening ports`, check the line CTIS, and select both the certificate and the private key you created earlier. By default, the CTIS port is 5013.



In your XiVO Client, in the menu *XiVO Client* → *Configure* → *Connection*, check the option `Encrypt connection` and adjust the server port if necessary.

> **Warning:** For now, there is no mechanism for strong authentification of the server. The connection is encrypted, but the identity of the server is not verified.

CTI profiles

The CTI profiles define which features are made available to a user. You can configure which profile will be used by a user in the menu *IPBX* → *PBX Settings* → *Users*:

You can also customize the default profiles or add new profiles in the menu *CTI Server* → *Profiles*:

Xlets

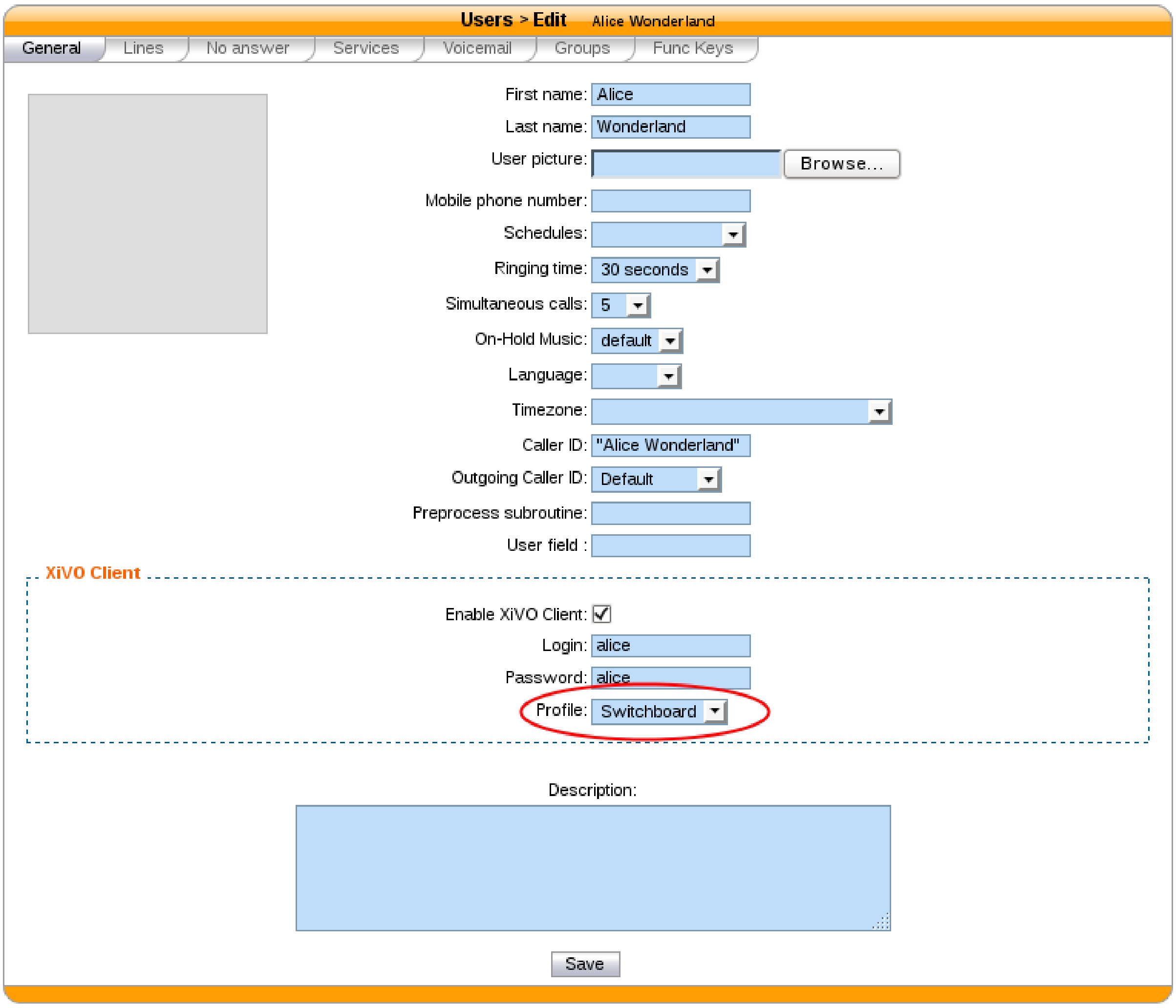
To choose which features are available to users using a profile, you have to select which *Xlets* will be available.

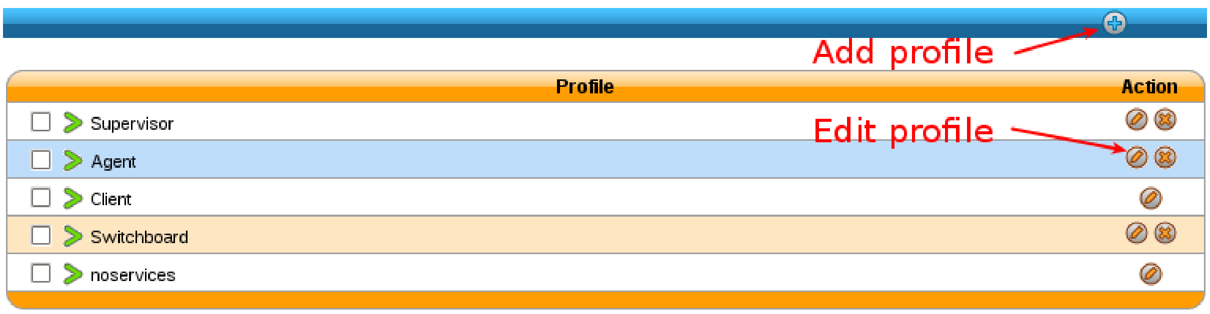
The Xlets are detailed in *Xlets*.

The *Position* attribute determines how the Xlets will be laid out:

- *dock* will display a Xlet in its own frame. This frame can have some options:
  - *Floating* means that the frame can be detached from the main window of the CTI Client.
  - *Closable* means that the Xlet can be hidden
  - *Movable* means that the Xlet can be moved (either inside the main window or outside)
  - *Scroll* means that the Xlet will display a scroll bar if the Xlet is too large.
- *grid* will display a Xlet inside the main window, and it will not be movable. Multiple *grid* Xlets will be laid out vertically (the second below the first).
- *tab* will display a Xlet inside a tab of the Xlet *Tabber*. Thus the Xlet *Tabber* is required and can't be in a *tab* position.

The *Number* attribute gives the order of the Xlets, beginning with 0. The order applies only to Xlets having the same *Position* attribute.


Users > Edit Alice Wonderland
General
Lines
No answer
Services
Voicemail
Groups
Func Keys
First name: Alice
Last name: Wonderland
User picture:
Browse...
Mobile phone number:
Schedules:
Ringing time: 30 seconds
Simultaneous calls: 5
On-Hold Music: default
Language:
Timezone:
Caller ID: "Alice Wonderland"
Outgoing Caller ID: Default
Preprocess subroutine:
User field :
XiVO Client
Enable XiVO Client:
Login: alice
Password: alice
Profile: Switchboard
Description:
Save


Add profile
Profile
Action
Edit profile
Supervisor
Agent
Client
Switchboard
noservices

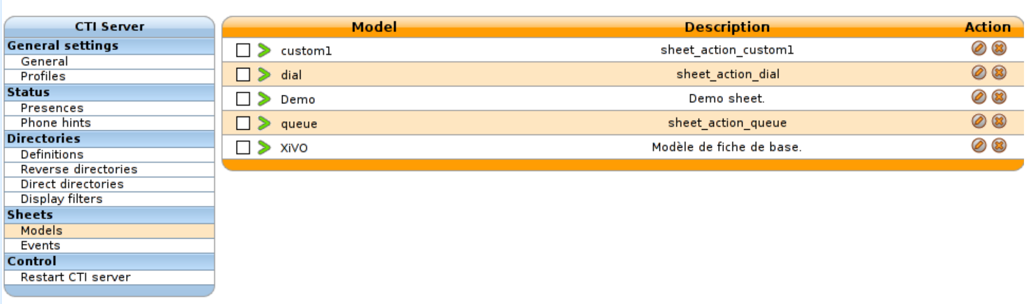
1.8.7 Display customer informations

Sheet Configuration

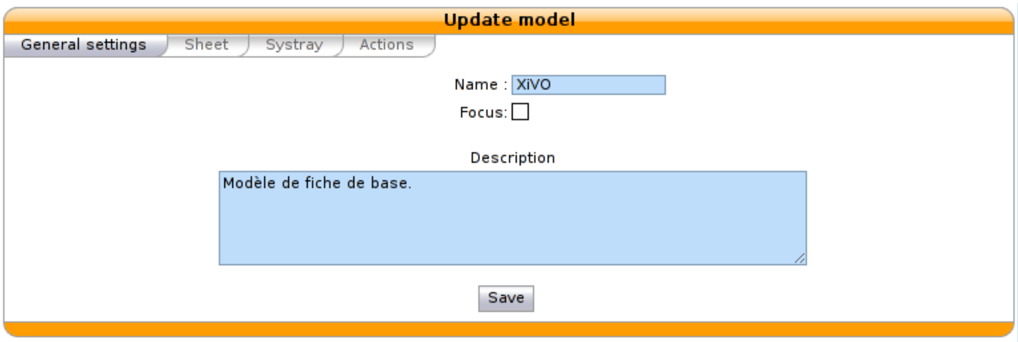
Sheets can be defined under *Services* → *CTI Server* → *Models* in the web interface. Once a sheet is defined, it has to be assigned to an event in the *Services* → *CTI Server* → *Events* menu.

Model The model contains the content of the displayed sheet.

Event Events are actions that trigger the defined sheet. A sheet can be assigned to many events. In that case, the sheet will be raised for each event.



General settings



You must give a name to your sheet to be able to select it later.

The `Focus` checkbox makes the XiVO Client pop up when the sheet is displayed, if the XiVO Client was hidden.

Sheets

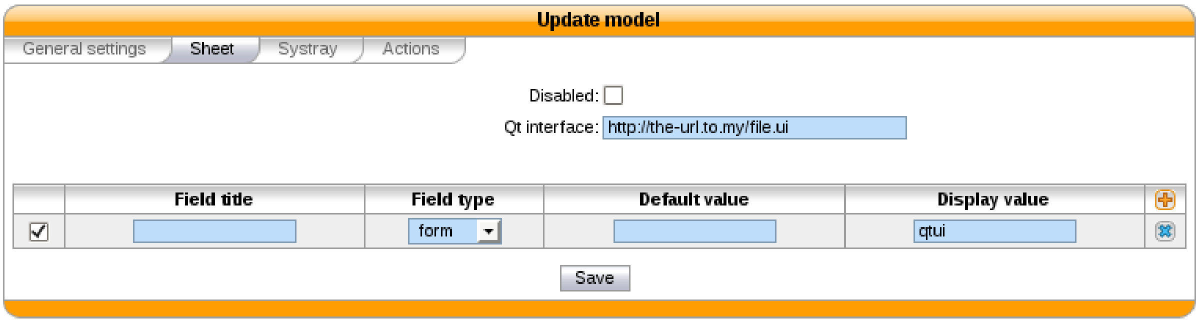
There are two different ways to configure the contents of the sheet:

- creating a custom sheet from the Qt designer. This gives you a total control on the layout of the information and allows you to save and process data entered during or after a call.
- listing the different fields and their content. The information will be automatically laid out in a linear fashion and will be read-only.

Custom sheet

Configuring the sheet The `Qt interface` field is the path to the UI file created by the Qt Designer. The path can either be a local file on your XiVO starting with `file://`, or a HTTP URL.

You must add a field with type `form` and display value `qtui` for the form to be displayed.



Create a custom sheet with Qt Designer The Qt Designer is part of the Qt development kit and is also available in the Qt Creator. They are available on the Qt project website.

Here is an example of a small form created with Qt Designer.

The Qt Designer screenshot.

> **Warning:** In Qt Designer, one must set 'vertical layout' on the top widget (right click on the top widget > Lay out > Vertical layout).

You can download the file generated by this example from Qt Designer: `example-form.ui`

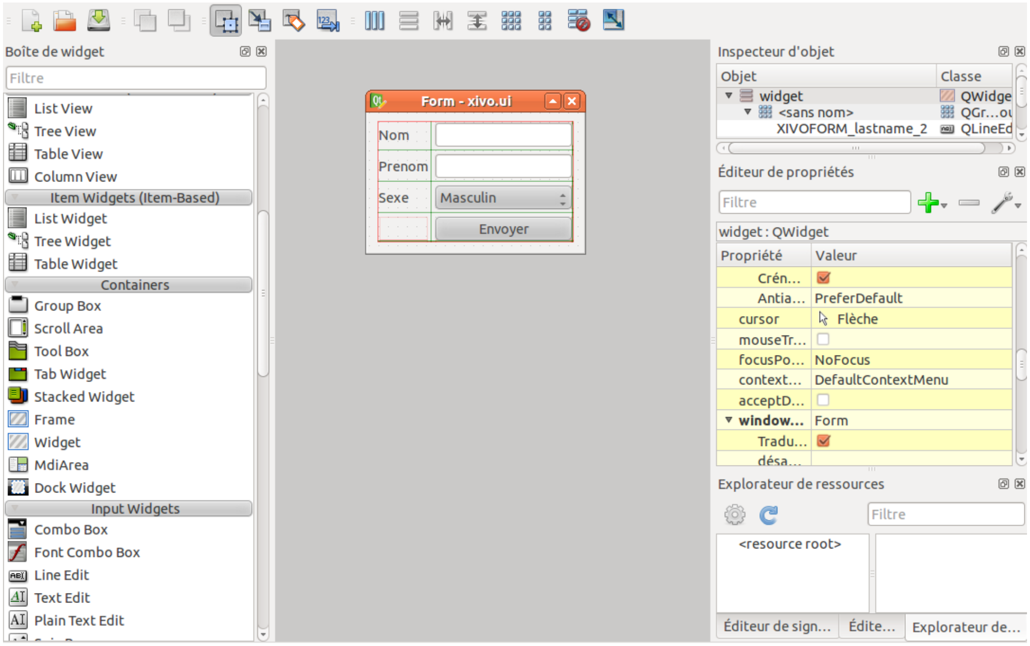
Text fields (QLineEdit, QLabel, QPlainTextEdit) can contain variables that will be substituted. See the *variable list* for more information.

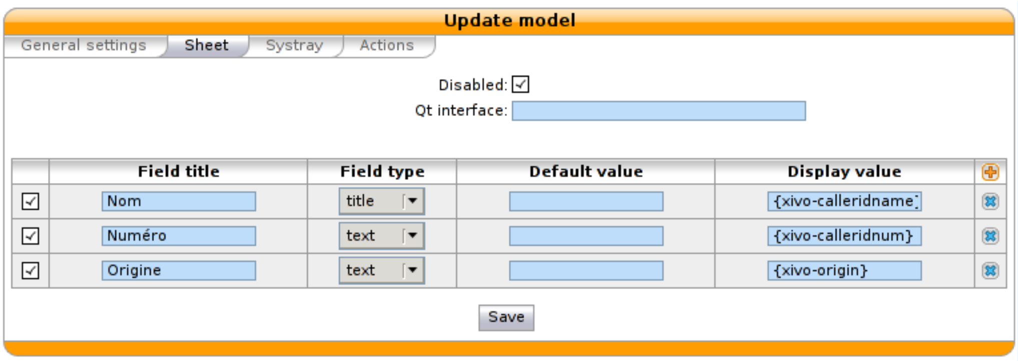
List of fields Default XiVO sheet example :

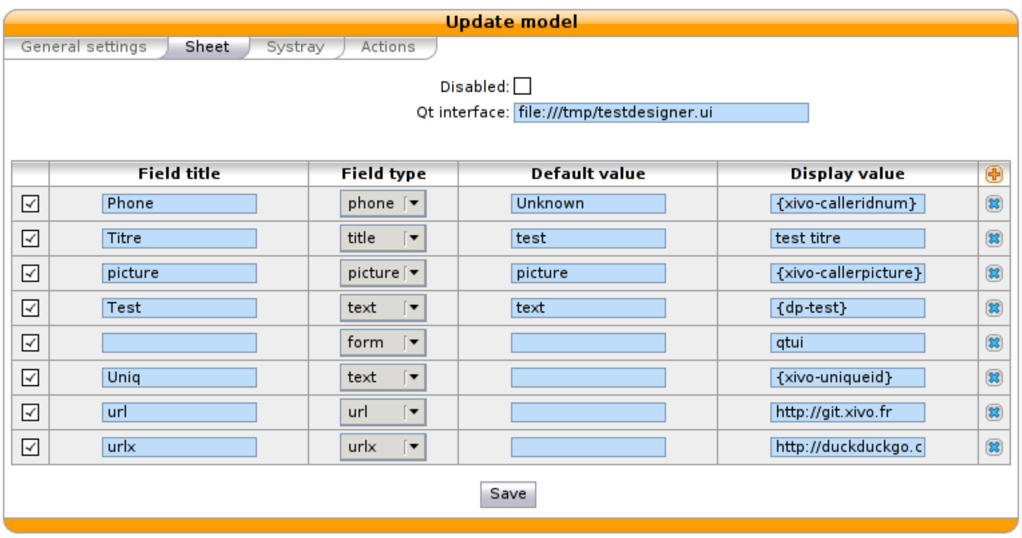
Example showing all kinds of fields:

Each field is represented by the following parameters :

- Field title : name of your line used as label on the sheet.
- Field type : define the type of field displayed on the sheet. Supported field types :
  - title : to create a title on your sheet
  - text : show a text
  - url : a simple url link, open your default browser.
  - urlx : an url button
  - picture : show a picture from an internal user in your sheet, you need to use {xivo-picture} variable.
  - phone : create a tel: link, you can click to call on your sheet.


Boîte de widget
Filtre
List View
Tree View
Table View
Column View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Scroll Area
Tool Box
Tab Widget
Stacked Widget
Frame
Widget
MdiArea
Dock Widget
Input Widgets
Combo Box
Font Combo Box
Line Edit
Text Edit
Plain Text Edit
Form - xivo.ui
Nom
Prenom
Sexe
Masculin
Envoyer
Inspecteur d'objet
Objet
Classe
widget
QWidge
<sans nom>
QGr...ou
XIVOFORM_lastname_2
QLineEd
Éditeur de propriétés
Filtre
widget : QWidget
Propriété
Valeur
Crén...
Antia...
PreferDefault
cursor
Flèche
mouseTr...
focusPo...
NoFocus
context...
DefaultContextMenu
acceptD...
window...
Form
Tradu...
désa...
Explorateur de ressources
Filtre
<resource root>
Éditeur de sign...
Édite...
Explorateur de...


Update model
General settings
Sheet
Systray
Actions
Disabled:
Qt interface:
Field title
Field type
Default value
Display value
Nom
title
{xivo-calleridname}
Numéro
text
{xivo-calleridnum}
Origine
text
{xivo-origin}
Save


Update model
General settings
Sheet
Systray
Actions
Disabled:
Qt interface: file:///tmp/testdesigner.ui
Field title
Field type
Default value
Display value
Phone
phone
Unknown
{xivo-calleridnum}
Titre
title
test
test titre
picture
picture
picture
{xivo-callerpicture}
Test
text
text
{dp-test}
form
qtui
Uniq
text
{xivo-uniqueid}
url
url
http://git.xivo.fr
urlx
urlx
http://duckduckgo.c
Save

  - form : show the form from an ui predefined. It's an xml ui. You need to define qtui in display format.
- Default value : if given, this value will be used when all substitutions in the display value field fail.
- Display value : you can define text, variables or both. See the *variable list* for more information.

Variables Three kinds of variables are available :

- *xivo-* prefix is reserved and set inside the CTI server:
  - *xivo-where* for sheet events, event triggering the sheet
  - *xivo-origin* place from where the lookup is requested (did, internal, forcelookup)
  - *xivo-direction* incoming or internal
  - *xivo-did* DID number
  - *xivo-calleridnum*
  - *xivo-calleridname*
  - *xivo-calleridrdnis* contains information whether there was a transfer
  - *xivo-calleridton* Type Of Network (national, international)
  - *xivo-calledidnum*
  - *xivo-calledidname*
  - *xivo-ipbxid* (*xivo-astid* in 1.1)
  - *xivo-directory* : for directory requests, it is the directory database the item has been found
  - *xivo-queuename* queue called
  - *xivo-agentnumber* agent number called
  - *xivo-date* formatted date string
  - *xivo-time* formatted time string, when the sheet was triggered
  - *xivo-channel* asterisk channel value (for advanced users)
  - *xivo-uniqueid* asterisk uniqueid value (for advanced users)
- *db-* prefixed variables are defined when the reverse lookup returns a result.

For example if you want to access to the reverse lookup full name, you need to define a field `fullname` in the directory definition, mapping to the full name field in your directory. The `{db-fullname}` will be replaced by the caller full name. Every field of the directory is accessible this way.

- *dp-* prefixed ones are the variables set through the dialplan (through UserEvent application)

For example if you want to access from the dialplan to a variable dp-test you need to add in your dialplan this line (in a subroutine):

```
UserEvent(dialplan2cti,UNIQUEID: ${UNIQUEID},CHANNEL: ${CHANNEL},VARIABLE: test,VALUE: "Salut
```

The `{dp-test}` displays Salut.

Sending informations during/after a call After showing a sheet, the XiVO Client can also send back information to XiVO for post-processing or archiving.

Here are the requirements:

- The sheet must contain a button named `save` to submit information
- Supported widgets:
  - QCalendarWidget

- QCheckBox
- QComboBox
- QDateEdit
- QDateTime
- QDateTimeEdit
- QDoubleSpinBox
- QLabel
- QLineEdit
- QList
- QPlainTextEdit
- QRadioButton
- QSpinBox
- QTimeEdit

• Fields must have their name starting with `XIVOFORM_`

If you want to send information that is not visible, you can make the widget invisible on the sheet:

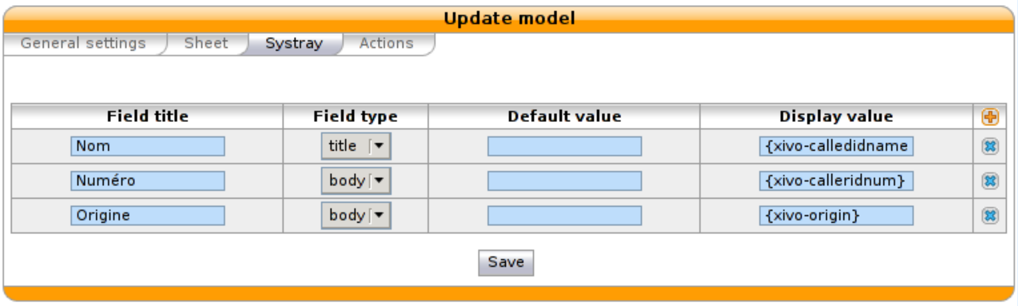
• change the maximumWidth or maximumHeight property to 0
• edit the `.ui` file and add the following property to the widget:

```
<property name="visible">
  <bool>false</bool>
</property>
```

When a CTI client submits a custom sheet, a *call_form_result* event is published on the event bus.

Systray

Mostly the same syntax as the sheet with less field types available (title, body). A Systray popup will display a single title (the last one added to the list of fields) and zero, one or more fields of type 'body'.



> **Warning:** The popup message on MacOSX works with Growl http://growl.info. We could get simple sheet popup to work using the free Growl Fork http://www.macupdate.com/app/mac/41038/growl-fork Note that this is not officially supported.

Actions

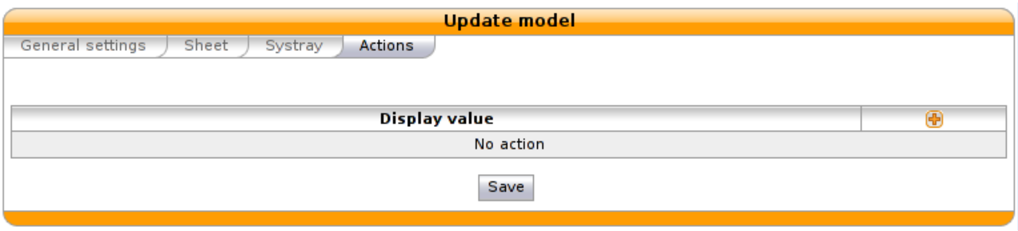
The action is for the xivo client, so if you configure an action, please be sure you understand it's executed *by the client*. You need to allow this action in the client configuration too (menu *XiVO Client -> Configure*, tab *Functions*, tick option *Customer Info* and in sub-tab *Customer Info* tick the option *Allow the Automatic Opening of URL*).

The field in this tab receives the URL that will be displayed in your browser. You can also use variable substitution in this field.

- `http://example.org/foo` opens the URL on the default browser
- `http://example.org/{xivo-did}` opens the URL on the default browser, after substituting the `{xivo-did}` variable. If the substitution fails, the URL will remain `http://example.org/{xivo-did}`, i.e. the curly brackets will still be present.
- `http://example.org/{xivo-did}?origin={xivo-origin}` opens the URL on the default browser, after substituting the variables. If at least one of the substitution is successful, the failing substitutions will be replaced by an empty string. For example, if `{xivo-origin}` is replaced by 'outcall' but `{xivo-did}` is not substituted, the resulting URL will be `http://example.org/?origin=outcall`
- `tcp://x.y.z.co.fr:4545/?var1=a1&var2=a2` connects to TCP port 4545 on x.y.z.co.fr, sends the string `var1=a1&var2=a2`, then closes
- `udp://x.y.z.co.fr:4545/?var1=a1&var2=a2` connects to UDP port 4545 on x.y.z.co.fr, sends the string `var1=a1&var2=a2`, then closes

Note: any string that would not be understood as an URL will be handled like and URL it is a process to launch and will be executed as it is written

For *tcp://* and *udp://*, it is a requirement that the string between */* and *?* is empty. An extension could be to define other serialization methods, if needed.



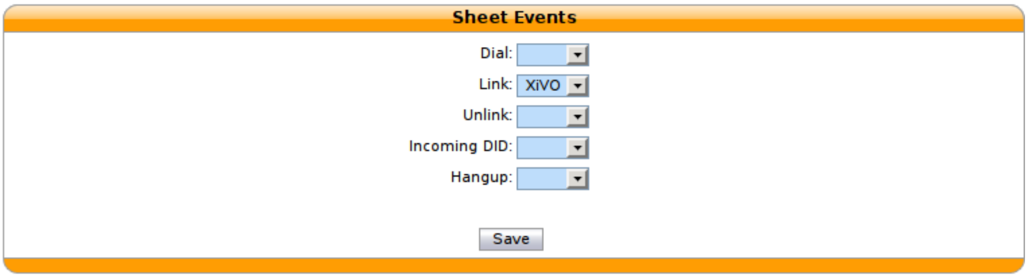
Event configuration

You can configure a sheet when a specific event is called. For example if you want to receive a sheet when an agent answers to a call, you can choose a sheet model for the Agent link event.

The following events are available :

- Dial: When a new call enters the queue (all the members of the queue will be notified, even if they are not logged)
- Link: When a user or agent answers a call
- Unlink: When a user or agent hangup a call received from a queue
- Incoming DID: Received a call in a DID
- Hangup: Hangup the call

The informations about a call are displayed via the XiVO Client on forms called sheets.



Example: Display a Web page when an agent answers a call

The first step is to assign the URL to a dialplan variable. Go in the *Services* → *IPBX* → *Configuration files* and create a new file called `setsheeturl.conf`. In this file, put the following:

```
[setsheeturl]
exten = s,1,NoOp(Starting Set Sheet URL)
same  = n,Set(SHEET_URL_CTI=http://documentation.xivo.fr)
same  = n,UserEvent(dialplan2cti,UNIQUEID: ${UNIQUEID},CHANNEL: ${CHANNEL},VARIABLE: mysheeturl,V
same  = n,Return()
```

You can replace `documentation.xivo.fr` by the URL you want.

The second step is to set the URL when the call is queued. To do that, we will use a preprocessing subroutine. This is configured in the queue configuration : go to *Services* → *Call center* → *Queues* and edit the queue. Set the field `Preprocessing subroutine` to `setsheeturl` (the same as above).

The third step is to configure the sheet to open the wanted URL. Go to *Services* → *CTI Server* → *Sheets* → *Models* and create a new sheet. Keep the default for everything except the Action tab, add a field and set it to `{dp-mysheeturl}` (the same as above).

The fourth and final step is to trigger the sheet when the agent answers the queued call. Go to *Services* → *CTI Server* → *Sheets* → *Events* and link the event `Agent linked` to the sheet you just created.

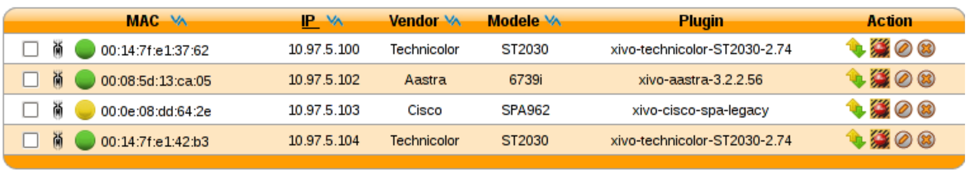
That's it, you can assign agents to your queue, log the agents and make them answer calls with the XiVO Client opened, and your browser should open the specified URL.

1.8.8 Devices

Synchronize a device

You first have to display list devices.

Figure 1.36: Click on synchronize button for a device.



MAC	IP	Vendor	Modele	Plugin	Action
00:14:7f:e1:37:62	10.97.5.100	Technicolor	ST2030	xivo-technicolor-ST2030-2.74	
00:08:5d:13:ca:05	10.97.5.102	Aastra	6739i	xivo-aastra-3.2.2.56	
00:0e:08:dd:64:2e	10.97.5.103	Cisco	SPA962	xivo-cisco-spa-legacy	
00:14:7f:e1:42:b3	10.97.5.104	Technicolor	ST2030	xivo-technicolor-ST2030-2.74	

Figure 1.37: List devices

You will see a pop-up to confirm synchronization Click on <ok> button.

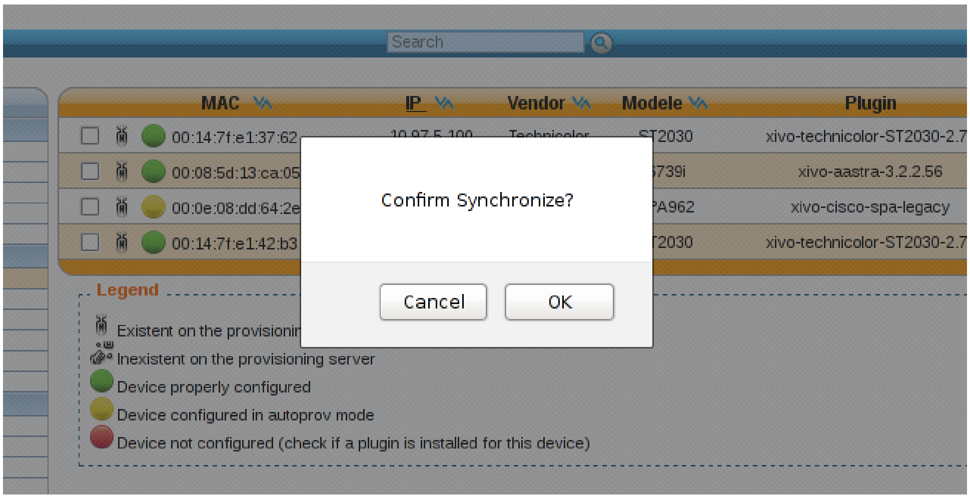


Figure 1.38: Alert confirm synchronize

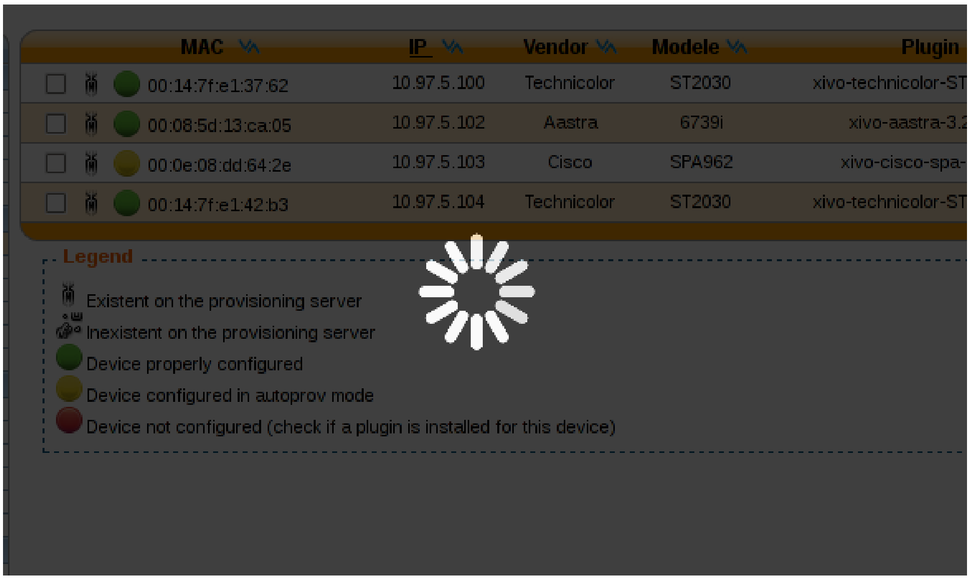


Figure 1.39: Request synchronisation processing

You must wait until the full synchronization process to determine the state back. This can take several seconds. It is important to wait and do nothing during that time.

If synchronization is successful, an information balloon green warn you of success.

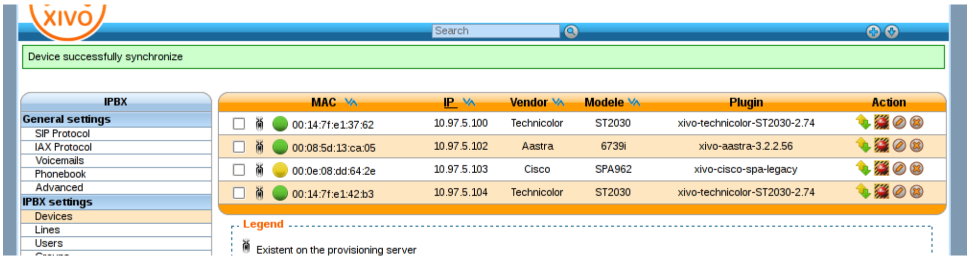


Figure 1.40: Device successfully synchronize

If synchronization fails, an information balloon red warning you of success.

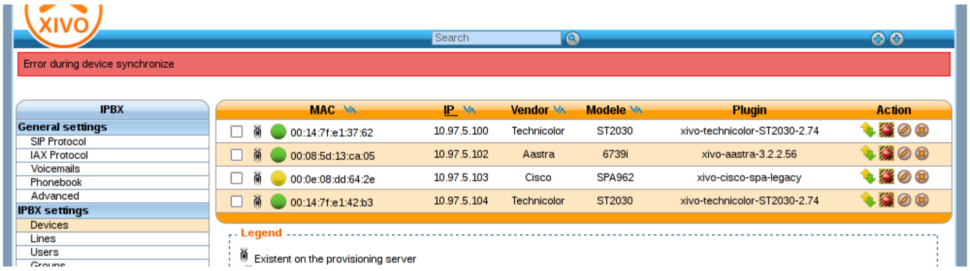


Figure 1.41: Error during device synchronize

Synchronize multiple devices

> **Warning:** By using multiple synchronization, the state of return will not be returned.

Select the devices you want to synchronize by checking the boxes.

Figure 1.42: Synchronize selected devices

A pop-up will emerge to request confirmation.

If synchronization mass was successfully sent to devices., an information balloon green warn you of success.

1.8.9 Directories

This page documents how to add and configure directories from custom sources. This does not include the configuration of *LDAP directories*, which are configured slightly differently.

Directories added from custom sources can be used for lookup via the *CTI Client* or for *reverse lookup* on incoming calls. The directory feature of phones do not use these data sources.

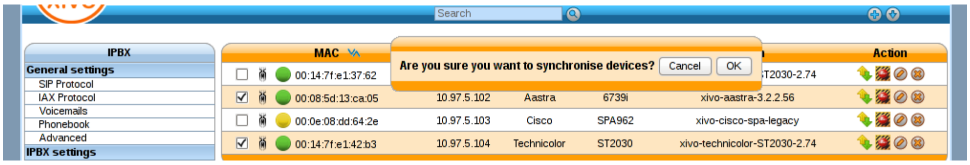


Figure 1.43: Synchronize selected devices confirmation

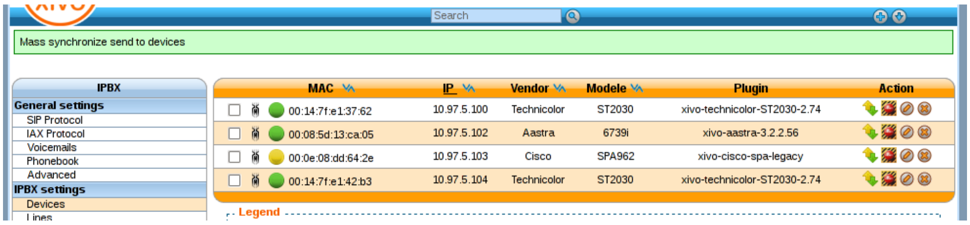


Figure 1.44: Mass synchronize request successfully send

Add a data source

You can add new data sources via the *Configuration* → *Management* → *Directories* page.

File directories

The source file of the directory must be in CSV format. You will be able to choose the headers and the separator in the next steps. For example, the file will look like:

```
title|firstname|lastname|displayname|society|mobilenumber|email
mr|Emmett|Brown|Brown Emmett|DMC|5555551234|emmet.brown@dmc.example.com
```

Example:

- *Directory name*: csv-phonebook
- *Type*: File
- *URI*: `/data/csv-phonebook.csv`

Web service directories

The data returned by the Web service must have the same format than the file directory. In the same way, you will be able to choose the headers and the separator in the next step.

Example:

- *Directory name*: ws-phonebook
- *Type*: Webservices
- *URI*: `http://example.org:8000/ws-phonebook`

Configure the access to the data source

Go in *Services* → *CTI Server* → *Directories* → *Definitions* and add a new directory definition.

- *URI*: your data source
- *Delimiter*: the field delimiter in your data source
- *Direct match*: the key used to match entries for direct lookup

- *Match reverse directories*: idem, but for reverse lookup
- *Mapped field*:
  - the *fieldname* is the identifier of the field. It will be used in the display filter, so look there if you want to use an existing one, or make it up if you want a custom display filter.
  - the *value* is the corresponding header of your data source.

File directories

For file directories, the *Direct match* and the *Match reverse directories* must be filled with the name of the column used to match entries.

For example, given you have the following CSV:

```
name|phone
John|5551234
```

And you want to do direct lookup on the `name` column and reverse lookup on the `phone` column, then you'll use:

- *Direct match*: name
- *Match reverse directories*: phone

Web service directories

For web service directories, the *Direct match* and the *Match reverse directories* must be filled with the name of the HTTP query parameter that will be used when doing the HTTP requests.

For example, given you have the following directory definition:

- *Direct match*: search
- *Match reverse directories*: phonesearch

When a direct lookup for "John" is performed, then the following HTTP request:

```
GET /ws-phonebook?search=John HTTP/1.1
```

is emitted. When a reverse lookup for "5551234" is performed, then the following HTTP request:

```
GET /ws-phonebook?phonesearch=5551234 HTTP/1.1
```

is emitted.

Note that the CSV returned by the Web service is not further processed.

Reverse lookup

To enable reverse lookup, you need to add an entry in *Mapped fields*:

- *Fieldname*: reverse
- *Value*: the header of your data source that you want to see as the caller ID on your phone on incoming calls

Configure the display of the data

Edit the default display filter or create your own in *Services → CTI Server → Directories → Display filters*.

Each line in the display filter will result in a header in your XiVO Client.

- *Field title* will be the text displayed in the header

- *Display format* is a format string, for example `{db-firstname} {db-lastname}`, where `{db-***}` will be replaced with the value from the data source. `***` is the identifier of the field configured in the directory definition, `'not'` the header of your data source.

Make your directory available

Go in *Services* → *CTI Server* → *Directories* → *Reverse/Direct directories*, select your display filter if needed and add the directory you just created.

You may have to restart the CTI Server or the AGI daemon to apply the change:

```
service xivo-ctid restart
service xivo-agid restart
```

1.8.10 Directed Pickup

Directed pickup allows a user to intercept calls made to another user.

For example, if a user with number 1001 is ringing, you can dial *81001 from your phone and it will intercept (i.e. pickup) the call to this user.

The extension prefix used to pickup calls can be changed via the *Services* → *IPBX* → *IPBX services* → *Extensions* page.

Custom Line Limitation

There is a case where directed pickup does not work, which is the following:

```
Given you have a user U with a line of type "customized"
Given this custom line is using DAHDI technology
Given this user is a member of group G
When a call is made to group G
Then you won't be able to intercept the call made to U by pressing *8<line number of U>
```

If you find yourself in this situation, you'll need to write a bit of dialplan.

For example, if you have the following:

- a user with a custom line with number 1001 in context default
- a custom line with interface `DAHDI/g1/5551234`

Then add the following, or similar:

```
[custom_lines]
exten = line1001,1,NoOp()
same  = n,Set(__PICKUPMARK=1001%default)
same  = n,Dial(DAHDI/g1/5551234)
same  = n,Hangup()
```

And do a `dialplan reload` in the asterisk CLI.

Then, edit the line of the user and change the interface value to `Local/line1001@custom_lines`

Note that you'll need to update your dialplan if you update the number of the line or the context.

1.8.11 Entities

> **Warning:** This feature is currently under development.

Purpose

In some cases, as the telephony provider, you want different independent organisation to have their telephony served by your XiVO, e.g. different departments using the same telephony infrastructure, but you do not want each organisation to see or edit the configuration of other organisations.

Configuration

In *Configuration* → *Entities*, you can create entities, one for each independant organisation.

In *Configuration* → *Users*, you can select an entity for each administrator.

Note: Once an entity is linked with an administrator, it can not be deleted. You have to unlink the entity from all administrator to be able to delete it.

1.8.12 Fax

Fax transmission

It's possible to send faxes from XiVO using the fax Xlet in the XiVO client.

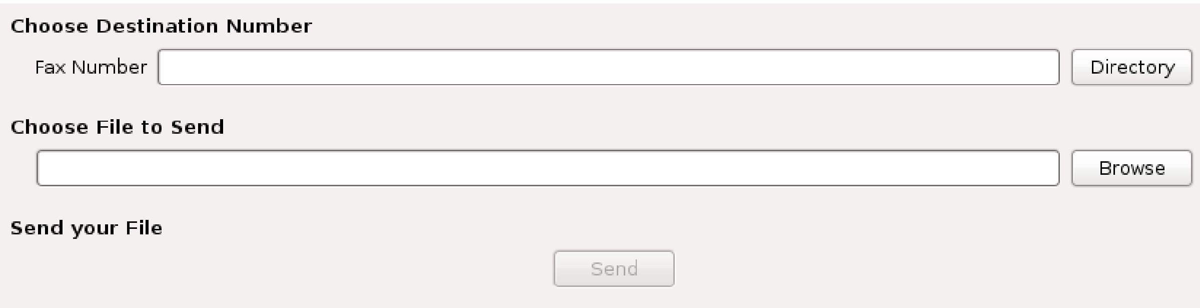


Figure 1.45: The fax Xlet in the XiVO Client

The file to send must be in PDF format.

Warning: Sending faxes is currently not supported if there is a network equipment that changes TCP port numbers or IP addresses, like a router doing NAT or NAPT, between the CTI client and the CTI server.

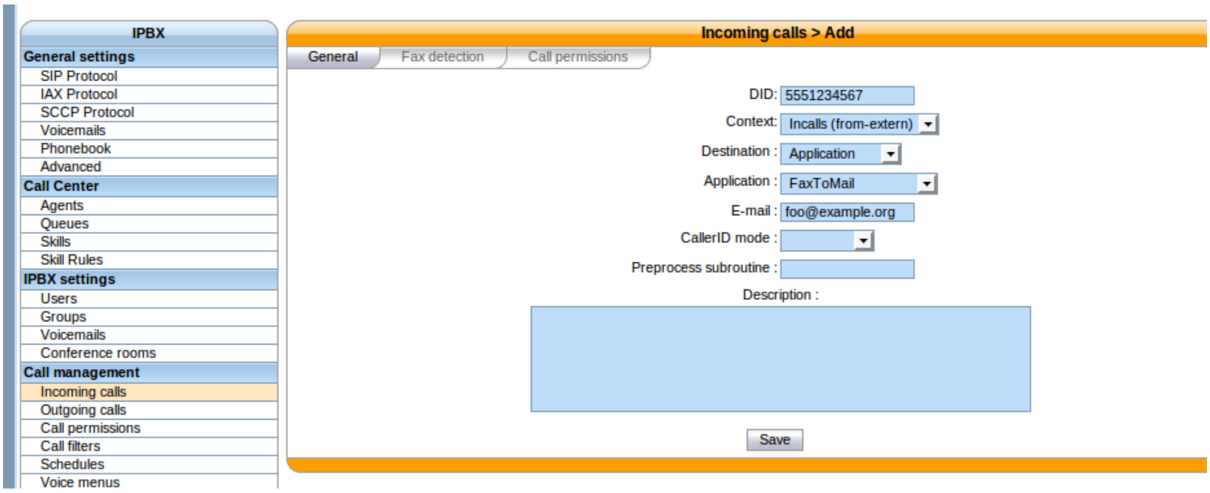
Fax reception

Adding a fax reception DID

If you want to receive faxes from XiVO, you need to add incoming calls definition with the *Application* destination and the *FaxToMail* application for every DID you want to receive faxes from.

This applies even if you want the action to be different from sending an email, like putting it on a FTP server. You'll still need to enter an email address in these cases even though it won't be used.

Note that, as usual when adding incoming call definitions, you must first define the incoming call range in the used context.



Changing the email body

You can change the body of the email sent upon fax reception by editing `/etc/xivo/mail.txt`.

The following variable can be included in the mail body:

- `%(dstnum)s`: the DID that received the fax

If you want to include a regular percent character, i.e. `%`, you must write it as `%%` in `mail.txt` or an error will occur when trying to do the variables substitution.

The `agid` service must be restarted to apply changes:

```
/etc/init.d/xivo-agid restart
```

Changing the email subject

You can change the subject of the email sent upon fax reception by editing `/etc/xivo/asterisk/xivo_fax.conf`.

Look for the `[mail]` section, and in this section, modify the value of the `subject` option.

The available variable substitution are the same as for the email body.

The `agid` service must be restarted to apply changes:

```
/etc/init.d/xivo-agid restart
```

Changing the email from

You can change the from of the email sent upon fax reception by editing `/etc/xivo/asterisk/xivo_fax.conf`.

Look for the `[mail]` section, and in this section, modify the value of the `email_from` option.

The `agid` service must be restarted to apply changes:

```
/etc/init.d/xivo-agid restart
```

Using the advanced features

The following features are only available via the `/etc/xivo/asterisk/xivo_fax.conf` configuration file. They are not available from the web-interface.

The configuration file has documentation embedded in it in the form of comments, so we recommend you reading them before editing the configuration file.

The way it works is the following:

- you first declare some backends, i.e. actions to be taken when a fax is received. A backend name looks like `mail`, `ftp_example_org` or `printer_office`.
- once your backends are defined, you can use them in your destination numbers. For example, when someone calls the DID 100, you might want the `ftp_example_org` and `mail` backend to be run, but otherwise, you only want the `mail` backend to be run.

Here's an example of a valid `/etc/xivo/asterisk/xivo_fax.conf` configuration file:

```
[general]
tiff2pdf = /usr/bin/tiff2pdf
mutt = /usr/bin/mutt
lp = /usr/bin/lp

[mail]
subject = FAX reception to %(dstnum)s
content_file = /etc/xivo/mail.txt
email_from = no-reply+fax@xivo.fr

[ftp_example_org]
host = example.org
username = foo
password = bar
directory = /foobar

[dstnum_default]
dest = mail

[dstnum_100]
dest = mail, ftp_example_org
```

The section named `dstnum_default` will be used only if no DID-specific actions are defined.

After editing `/etc/xivo/asterisk/xivo_fax.conf`, you need to restart the agid server for the changes to be applied:

```
$ /etc/init.d/xivo-agid restart
```

Using the FTP backend The FTP backend is used to send a PDF version of the received fax to an FTP server.

An FTP backend is always defined in a section beginning with the `ftp` prefix. Here's an example for a backend named `ftp_example_org`:

```
[ftp_example_org]
host = example.org
username = foo
password = bar
directory = /foobar
```

The `directory` option is optional and if not specified, the document will be put in the user's root directory.

The uploaded file are named like `${XIVO_SRCNUM}-${EPOCH}.pdf`.

Using the printer backend To use the printer backend, you must have the `cups-client` package installed on your XiVO:

```
$ apt-get install cups-client
```

The printer backend uses the `lp` command to print faxes.

A printer backend is always defined in a section beginning with the `printer` prefix. Here's an example for a backend named `printer_office`:

```
[printer_office]
name = office
convert_to_pdf = 1
```

When a fax will be received, the system command `lp -d office <faxfile>` will be executed.

The `convert_to_pdf` option is optional and defaults to 1. If it is set to 0, the TIFF file will not be converted to PDF before being printed.

Warning: You need a CUPS server set up somewhere on your network.

Using the mail backend By default, a mail backend named `mail` is defined. You can define more mail backends if you want. Just look what the default mail backend looks like.

Using the log backend There's also a log backend available, which can be used to write a line to a file every time a fax is received.

Fax detection

XiVO **does not currently support Fax Detection**. A workaround is described in the *Fax detection* section.

Using analog gateways

XiVO is able to provision Linksys SPA2102, SPA3102 and SPA8000 analog gateways which can be used to connect fax equipments. This section describes the creation of custom template *for SPA3102* which modifies several parameters.

Note: Be aware that most of the parameters are or could be country specific, i.e. :

- Preferred Codec,
- FAX Passthru Codec,
- RTP Packet Size,
- RTP-Start-Loopback Codec,
- Ring Waveform,
- Ring Frequency,
- Ring Voltage,
- FXS Port Impedance

1. Create a custom template for the SPA3102 base template:

   ```
   cd /var/lib/xivo-provd/plugins/xivo-cisco-spa3102-5.1.10/var/templates/
   cp ../../templates/base.tpl .
   ```

2. Add the following content before the `</flat-profile>` tag:

   ```
   <!-- CUSTOM TPL - for faxes - START -->

   {% for line_no, line in sip_lines.iteritems() %}
   <!-- Dial Plan: L{{ line_no }} -->
   ```

```
<Dial_Plan_{{ line_no }}_ ua="na">([x*#].)</Dial_Plan_{{ line_no }}_>

<Call_Waiting_Serv_{{ line_no }}_ ua="na">No</Call_Waiting_Serv_{{ line_no }}_>
<Three_Way_Call_Serv_{{ line_no }}_ ua="na">No</Three_Way_Call_Serv_{{ line_no }}_>

<Preferred_Codec_{{ line_no }}_ ua="na">G711a</Preferred_Codec_{{ line_no }}_>
<Silence_Supp_Enable_{{ line_no }}_ ua="na">No</Silence_Supp_Enable_{{ line_no }}_>
<Echo_Canc_Adapt_Enable_{{ line_no }}_ ua="na">No</Echo_Canc_Adapt_Enable_{{ line_no }}_>
<Echo_Supp_Enable_{{ line_no }}_ ua="na">No</Echo_Supp_Enable_{{ line_no }}_>
<Echo_Canc_Enable_{{ line_no }}_ ua="na">No</Echo_Canc_Enable_{{ line_no }}_>
<Use_Pref_Codec_Only_{{ line_no }}_ ua="na">yes</Use_Pref_Codec_Only_{{ line_no }}_>
<DTMF_Tx_Mode_{{ line_no }}_ ua="na">Normal</DTMF_Tx_Mode_{{ line_no }}_>

<FAX_Enable_T38_{{ line_no }}_ ua="na">Yes</FAX_Enable_T38_{{ line_no }}_>
<FAX_T38_Redundancy_{{ line_no }}_ ua="na">1</FAX_T38_Redundancy_{{ line_no }}_>
<FAX_Passthru_Method_{{ line_no }}_ ua="na">ReINVITE</FAX_Passthru_Method_{{ line_no }}_>
<FAX_Passthru_Codec_{{ line_no }}_ ua="na">G711a</FAX_Passthru_Codec_{{ line_no }}_>
<FAX_Disable_ECAN_{{ line_no }}_ ua="na">yes</FAX_Disable_ECAN_{{ line_no }}_>
<FAX_Tone_Detect_Mode_{{ line_no }}_ ua="na">caller or callee</FAX_Tone_Detect_Mode_{{ line_n

<Network_Jitter_Level_{{ line_no }}_ ua="na">very high</Network_Jitter_Level_{{ line_no }}_>
<Jitter_Buffer_Adjustment_{{ line_no }}_ ua="na">disable</Jitter_Buffer_Adjustment_{{ line_no
{% endfor %}

<!-- SIP Parameters -->
<RTP_Packet_Size ua="na">0.020</RTP_Packet_Size>
<RTP-Start-Loopback_Codec ua="na">G711a</RTP-Start-Loopback_Codec>

<!-- Regional parameters -->
<Ring_Waveform ua="rw">Sinusoid</Ring_Waveform> <!-- options: Sinusoid/Trapezoid -->
<Ring_Frequency ua="rw">50</Ring_Frequency>
<Ring_Voltage ua="rw">85</Ring_Voltage>

<FXS_Port_Impedance ua="na">600+2.16uF</FXS_Port_Impedance>
<Caller_ID_Method ua="na">Bellcore(N.Amer,China)</Caller_ID_Method>
<Caller_ID_FSK_Standard ua="na">bell 202</Caller_ID_FSK_Standard>

<!-- CUSTOM TPL - for faxes - END -->
```

3. Reconfigure the devices with:

```
provd_pycli -c 'devices.using_plugin("xivo-cisco-spa3102-5.1.10").reconfigure()'
```

4. Then reboot the devices:

```
provd_pycli -c 'devices.using_plugin("xivo-cisco-spa3102-5.1.10").synchronize()'
```

Most of this template can be copy/pasted for a SPA2102 or SPA8000.

Using a SIP Trunk

Fax transmission, to be successful, *MUST* use G.711 codec. Fax streams cannot be encoded with lossy compression codecs (like G.729a).

That said, you may want to establish a SIP trunk using G.729a for all other communications to save bandwith. Here's a way to be able to receive a fax in this configuration.

Note: There are some prerequisites:

- your SIP Trunk must offer both G.729a and G.711 codecs
- your fax users must have a customized outgoing calleridnum (for the codec change is based on this variable)

1. We assume that outgoing call rules and fax users with their DID are created
2. Create the file `/etc/asterisk/extensions_extra.d/fax.conf` with the following content:

```
;; For faxes :
; The following subroutine forces inbound and outbound codec to alaw.
; For outbound codec selection we must set the variable with inheritance.
; Must be set on each Fax DID
[pre-incall-fax]
exten = s,1,NoOp(### Force alaw codec on both inbound (operator side) and outbound (analog gw
exten = s,n,Set(SIP_CODEC_INBOUND=alaw)
exten = s,n,Set(__SIP_CODEC_OUTBOUND=alaw)
exten = s,n,Return()

; The following subroutine forces outbound codec to alaw based on outgoing callerid number
; For outbound codec selection we must set the variable with inheritance.
; Must be set on each outgoing call rule
[pre-outcall-fax]
exten = s,1,NoOp(### Force alaw codec if caller is a Fax ###)
exten = s,n,GotoIf($["${CALLERID(num)}" = "0112697845"]?alaw:)
exten = s,n,GotoIf($["${CALLERID(num)}" = "0112697846"]?alaw:end)
exten = s,n(alaw),Set(__SIP_CODEC_OUTBOUND=alaw)
exten = s,n(end),Return()
```

3. For each Fax users' DID add the following string in the `Preprocess subroutine` field:

```
pre-incall-fax
```

4. For each Outgoing call rule add the the following string in the `Preprocess subroutine` field:

```
pre-outcall-fax
```

1.8.13 Graphics

The Services/Graphics section gives a historical overview of a XiVO system's activity based on snapshots recorded every 5 minutes. Graphics are available for the following resources :

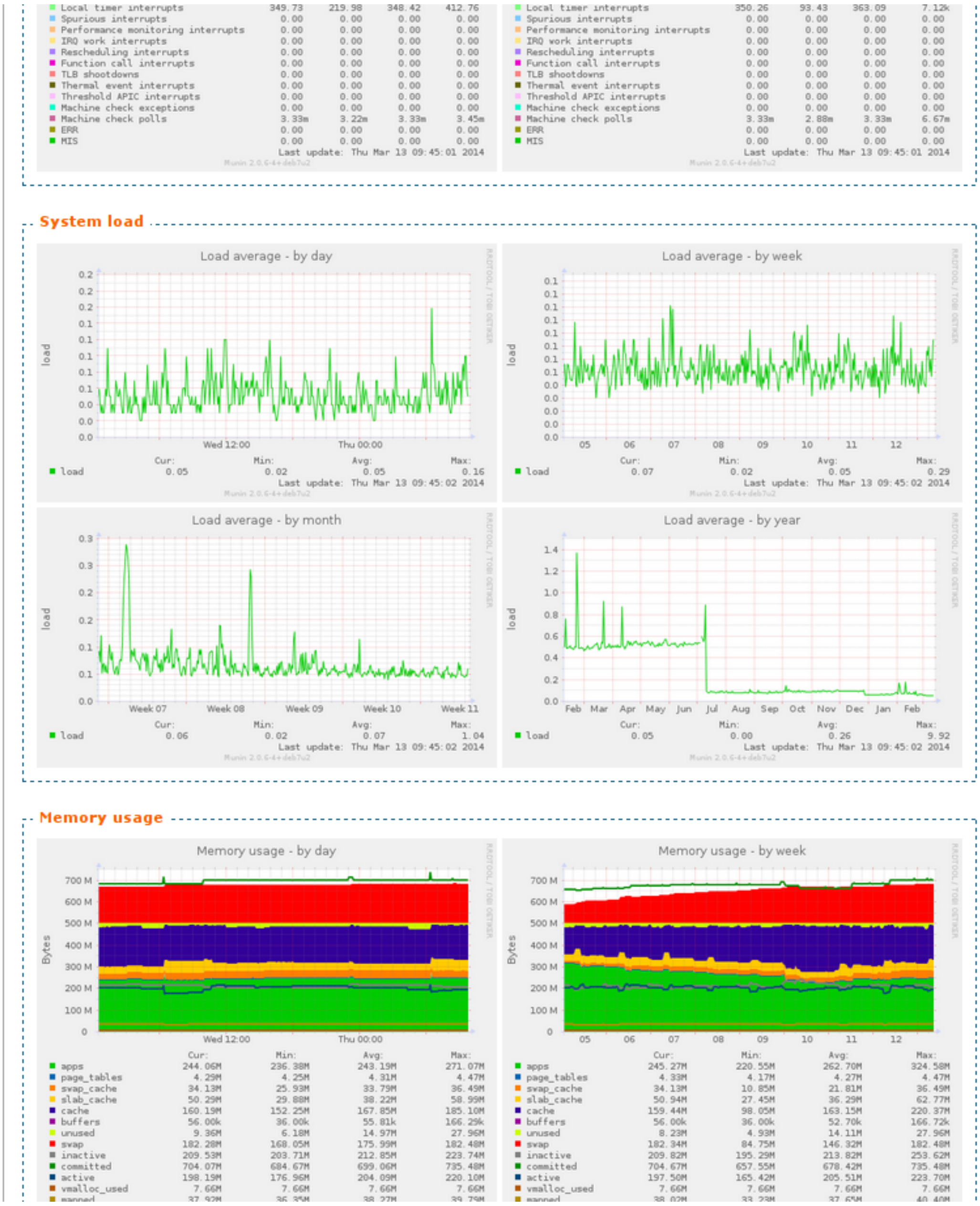
- CPU
- Entropy
- Interruptions
- IRQ Stats
- System Load
- Memory Usage
- Open Files
- Open Inodes
- Swap Usage

Each section is presented as a series of 4 graphics : daily, weekly, monthly and yearly history. Each graphic can be clicked on to zoom. All information presented is read only.

1.8.14 Groups

Groups are used to be able to call a set or users.

Group name cannot be `general` reserved in asterisk configuration.


System load
Load average - by day
Load average - by week
Load average - by month
Load average - by year
Memory usage
Memory usage - by day
Memory usage - by week

1.8.15 Group Pickup

Pickup groups allow users to intercept calls directed towards other users of the group. This is done either by dialing a special extension or by pressing a function key.

The group pickup is limited to 64 groups.

Quick Summary

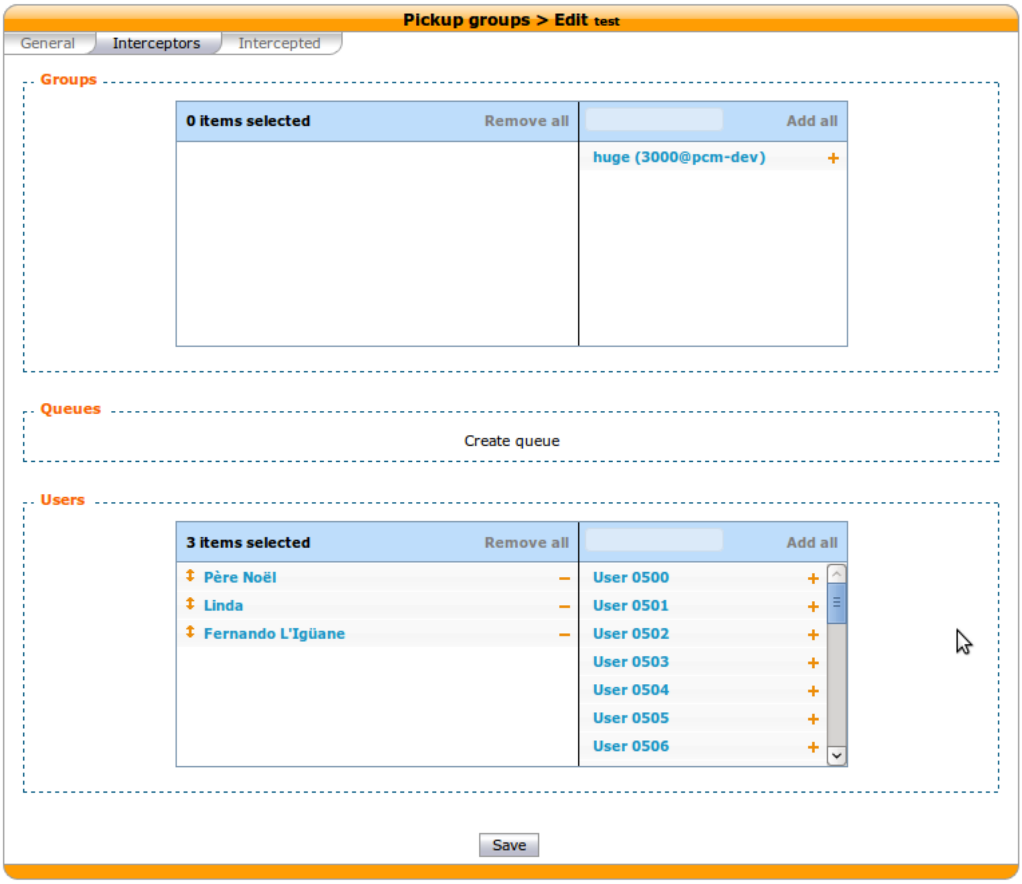
In order to be able to use group pickup you have to:

- Create a pickup group
- Enable an extension to intercept calls
- Add a function key to interceptors

Creating a Pickup Group

Pickup groups can be create in the *Services* → *Call management* → *Call pickups* page.

In the *general* tab, you can define a name and a description for the pickup group. In the *Interceptors* tab, you can define a list of users, groups or queues that can intercept calls. In the *Intercepted* tab, you can define a list of users, groups or queues that can be intercepted.



Enabling an Interception Extension

The pickup extension can be defined in the *Services* → *Extensions* page.

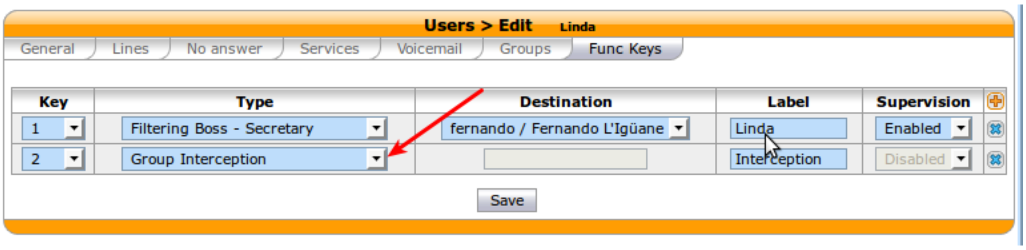
The extension used by group pickup is called *Group interception* it's default value is **8*.

> **Warning:** The extension must be enabled even if a function key is used.

Adding a Function Key to an Interceptor

To assign a function to an interceptor, go to *Services* → *Users*, edit an interceptor and go to the *Func Keys* tab.

Add a new function key of type *Group Interception* and save.



1.8.16 Server/Hardware

This section describes how to configure the telephony hardware on a XiVO server.

> **Note:** Currently XiVO support only Digium Telephony Interface cards

The configration process is the following :

1. Load the right DAHDI modules,
2. Configure the echo-canceller,
3. Configure your card and the associated span in asterisk.

At the end of this page you will also find some general notes and DAHDI.

Load the correct DAHDI modules

- Know which card is in your server:

You can see which cards are detected by issuing the `dahdi_hardware` command:

```
dahdi_hardware
pci:0000:05:0d.0     wcb4xxp+     d161:b410 Digium Wildcard B410P
pci:0000:05:0e.0     wct4xxp+     d161:0205 Wildcard TE205P (4th Gen)
```

- Then you have to comment all the unused modules in `/etc/dahdi/modules`.

For example, if you have one B410P and one TE205P you should comment every modules in `/etc/dahdi/modules` except:

```
wcb4xxp
wct4xxp
```

- **If this is a TE13X card** (`wcte13xp` module) you **MUST** create a configuration file to set the line mode as E1 (or T1).

Contrarily to other cards there is no jumper to change the line mode. The configuration below sets the card in E1 mode:

```
cat << EOF > /etc/modprobe.d/xivo-wcte13xp.conf
# set wcte13xp cards in E1/T2 mode
options wcte13xp default_linemode=e1
EOF
```

- Then, restart dahdi:

  ```
  xivo-service restart
  ```

Hardware Echo-cancellation

It is *recommended* to use telephony cards with an hardware echo-canceller module.

> **Warning:** with **TE13X** cards, you **MUST** install the echo-canceller firmware. Otherwise the card won't work properly.

Hardware Echo-cancellation Module

If you have an hardware echo-canceller module you **HAVE TO** install its firmware. This can be achieved via the `xivo-fetchfw` tool :

- Know which firmware you need :

The simplest way is to restart dahdi and then to lookup in the dmesg which firmware does DAHDI request at startup:

```
dmesg |grep firmware
[    7.781192] wct4xxp 0000:05:0e.0: firmware: requesting dahdi-fw-oct6114-064.bin
```

Otherwise you can also issue (with DAHDI >= 2.5.0) the `cat /proc/dahdi/1` command (assuming that the span 1 is a PRI port) and you should see lines containing something like `EC: VPMOCT64` which tells you the echo-canceller module you have:

```
cat /proc/dahdi/1
Span 1: TE2/0/1 "T2XXP (PCI) Card 0 Span 1" HDB3/CCS ClockSource

1 TE2/0/1/1 Clear (In use) (EC: VPMOCT064 - INACTIVE)
.....................................................
```

- Use xivo-fetchfw to find the name of the package :

You can search for `digium` occurences in the available packages:

```
xivo-fetchfw search digium
```

- Install the package :

In our example, we install the package named `digium-oct6114-064`:

```
xivo-fetchfw install digium-oct6114-064
```

Get help on xivo-fetchfw:

```
xivo-fetchfw -h
```

Activate the Hardware Echo-cancellation

To use the hardware echo-canceller of the card you must activate it in `/etc/asterisk/chan_dahdi.conf` file:

```
echocancel = 1
```

Use the Hardware Echo-canceller for DTMF detection

If you have an hardware echo-canceller it can be used to detect the DTMF.

Create the file `/etc/modprobe.d/xivo-hwec-dtmf.conf` with the following content (replace the <dahdi_module_name> word by the DAHDI module name):

```
options <dahdi_module_name> vpmdtmfsupport=1
```

Thus, for a Digium card which uses the `wct4xxp` module, the content of the file will be:

```
options wct4xxp vpmdtmfsupport=1
```

Note: You MUST restart dahdi for the new configuration to be enabled

Warning: Don't forget the extension `.conf` for the filename. Otherwise it won't be taken into account.

BRI card configuration

Verifications

Verify that the `wcb4xxp` module is uncommented in `/etc/dahdi/modules`.

If it wasn't, do again the step *Load the correct DAHDI modules*.

Generate DAHDI configuration

Issue the command:

```
dahdi_genconf
```

Warning: it will erase all existing configuration in `/etc/dahdi/system.conf` and `/etc/asterisk/dahdi-channels.conf` files !

Configure

- Modify the `/etc/dahdi/system.conf` file:
  - Check the span numbering,
  - If needed change the clock source,
  - Usually (at least in France) you should remove the `crc4`,

  Following is **an example** `/etc/dahdi/system.conf` file for a B410P 4 ports for French network (check the comments and see the */etc/dahdi/system.conf* section !):

  ```
  # Span 1: B4/0/1 "B4XXP (PCI) Card 0 Span 1" (MASTER) RED
  # span=1 (this is the first span),
  #       1 (this is the primary clock source)
  #       0 (-)
  #       ccs (use ccs framing)
  #       ami (use ami coding )
  span=1,1,0,ccs,ami
  ```

```
# termtype: te
bchan=1-2
hardhdlc=3
echocanceller=mg2,1-2

# Span 2: B4/0/2 "B4XXP (PCI) Card 0 Span 2" RED
span=2,2,0,ccs,ami
# termtype: te
bchan=4-5
hardhdlc=6
echocanceller=mg2,4-5

# Span 3: B4/0/3 "B4XXP (PCI) Card 0 Span 3" RED
span=3,3,0,ccs,ami
# termtype: te
bchan=7-8
hardhdlc=9
echocanceller=mg2,7-8

# Span 4: B4/0/4 "B4XXP (PCI) Card 0 Span 4" RED
# span=4 (this is the fourth span),
#       0 (won't use this span as a sync source)
#       0 (-)
#       ccs (use ccs framing)
#       ami (use ami coding )
span=4,0,0,ccs,ami
# termtype: nt
bchan=10-11
hardhdlc=12
echocanceller=mg2,10-11
```

- Modify the `/etc/asterisk/dahdi-channels.conf` file :
  - remove the unused lines like:

    ```
    context = default
    group = 63
    ```

  - Change the `context` lines if needed,
  - The `signaling` should be one of `{bri_net,bri_cpe,bri_net_ptmp,bri_cpe_ptmp}`.

  Following is **an example** `/etc/asterisk/dahdi-channels.conf` file for a B410P 4 ports for French network (check the comments and the */etc/asterisk/dahdi-channels.conf* section !):

```
; Span 1: B4/0/1 "B4XXP (PCI) Card 0 Span 1" (MASTER) RED
group=0,11              ; belongs to group 0 and 11
context=from-extern     ; incoming call to this span will be sent in 'from-extern' context
switchtype = euroisdn
signalling = bri_cpe    ; use 'bri_cpe' signaling
channel => 1-2          ; the above configuration applies to channels 1 and 2

; Span 2: B4/0/2 "B4XXP (PCI) Card 0 Span 2" RED
group=0,12
context=from-extern
switchtype = euroisdn
signalling = bri_cpe
channel => 4-5

; Span 3: B4/0/3 "B4XXP (PCI) Card 0 Span 3" RED
group=0,13
context=from-extern
switchtype = euroisdn
signalling = bri_cpe
```

```
channel => 7-8

; Span 4: B4/0/4 "B4XXP (PCI) Card 0 Span 4" RED
group=1,14              ; belongs to groups 1 and 14
context=default         ; incoming call to this span will be sent in 'defaul' context
switchtype = euroisdn
signalling = bri_net    ; use 'bri_net' signaling
channel => 10-11        ; the above configuration applies to channels 10 and 11
```

Special cases

Here are some special cases where you might need to modify the default options :

- if your telecom operator brings layer 1 down when the line is idle, you should add the following option in `/etc/asterisk/chan_dahdi.conf` and restart asterisk (works with XiVO 12.20 and above):

  ```
  layer2_persistence=keep_up
  ```

PRI card configuration

Verifications

Verify that one of the `{wct1xxp,wcte11xp,wcte12xp,wcte13xp,wct4xxp}` module is uncommented in `/etc/dahdi/modules` depending on the card you installed in your server.

If it wasn't, do again the step *Load the correct DAHDI modules*

> **Warning:** **TE13XP** cards :
> - these cards need a specific dahdi module configuration. See *Load the correct DAHDI modules* paragraph,
> - you **MUST** install the correct echo-canceller firmware to be able to use these cards. See *Hardware Echo-cancellation* paragraph.

Generate DAHDI configuration

Issue the command:

```
dahdi_genconf
```

> **Warning:** it will erase all existing configuration in `/etc/dahdi/system.conf` and `/etc/asterisk/dahdi-channels.conf` files !

Configure

- Modify the `/etc/dahdi/system.conf` :
- Check the span numbering,
- If needed change the clock source,
- Usually (at least in France) you should remove the `crc4`,
- Modify the `/etc/asterisk/dahdi-channels.conf` file :
- remove the unused lines like:

  ```
  context = default
  group = 63
  ```

- Change the `context` lines if needed,
- The `signaling` should be one of `{pri_net,pri_cpe}`.

Sync cable You can link several PRI Digium card between themselves with a sync cable to share the exact same clock.

If you do this, you need to:

- use the coding wheel on the Digium cards to give them an order of recognition in DAHDI/Asterisk (see Digium_telephony_cards_support),
- daisy-chain the cards with a sync cable (see Digium_telephony_cards_support),
- load the DAHDI module with the `timingcable=1` option.

Create `/etc/modprobe.d/xivo-timingcable.conf` file and insert the line:

```
options <module> timingcable=1
```

Where <module> is the DAHDI module name of your card (e.g. wct4xxp for a TE205P).

Analog card configuration

Verifications

Verify that one of the `{wctdm,wctdm24xxp}` module is uncommented in `/etc/dahdi/modules` depending on the card you installed in your server.

If it wasn't, do again the step *Load the correct DAHDI modules*

Generate DAHDI configuration

Issue the command:

```
dahdi_genconf
```

> **Warning:** it will erase all existing configuration in `/etc/dahdi/system.conf` and `/etc/asterisk/dahdi-channels.conf` files !

Configure

- With **FXS** modules :

Create file `/etc/modprobe.d/xivo-tdm`:

```
options <module> fastringer=1 boostringer=1
```

Where <module> is the DAHDI module name of your card (e.g. wctdm for a TDM400P).

- With **FXO** modules:

Create file `/etc/modprobe.d/xivo-tdm`:

```
options <module> opermode=FRANCE
```

Where <module> is the DAHDI module name of your card (e.g. wctdm for a TDM400P).

1. Modify the `/etc/dahdi/system.conf` :
2. Check the span numbering,
3. Modify the `/etc/asterisk/dahdi-channels.conf` file :

- remove the unused lines like:

```
context = default
group = 63
```

- Change the `context` lines if needed

Voice Compression Card configuration

Here's how to install a Digium TC400M card (used for G.729a and/or G.723.1 codecs) :

- Verify that the `wctc4xxp` module is uncommented in `/etc/dahdi/modules`. If it wasn't, do again the step *Load the correct DAHDI modules*.
- install the card firmware:

```
xivo-fetchfw install digium-tc400m
```

- comment out the following line in `/etc/asterisk/modules.conf`:

```
noload = codec_dahdi.so
```

- restart asterisk:

```
/etc/init.d/asterisk restart
```

- depending on the codec you want to transcode, you can modify the `mode` parameter of the module by creating a file in `/etc/modprobe.d/`. This parameter can take the following value :
- mode = mixed : this the default value which activates transcoding for 92 channels in G.729a or G.723.1 (5.3 Kbit and 6.3 Kbit)
- mode = g729 : this option activates transcoding for 120 channels in G.729a
- mode = g723 : this option activates transcoding for 92 channels in G.723.1 (5.3 Kbit et 6.3 Kbit)

Example:

```
cat << EOF > /etc/modprobe.d/xivo-transcode.conf
options wctc4xxp mode=g729
EOF
```

After having applied the configuration (see Apply configuration section) you can verify that the card is correctly seen by asterisk with the `transcoder show` CLI command - this command should show the encoders/decoders registered by the TC400 card:

```
*CLI> transcoder show
0/0 encoders/decoders of 120 channels are in use.
```

Apply configuration

When done, you have to restart asterisk and dahdi:

```
/etc/init.d/monit stop
/etc/init.d/asterisk stop
/etc/init.d/dahdi stop
/etc/init.d/dahdi start
/etc/init.d/asterisk start
/etc/init.d/monit start
```

Check IRQ misses

It's always useful to verify if there isn't any *missed IRQ* problem with the cards.

Check:

```
cat /proc/dahdi/<span number>
```

If the *IRQ misses* counter increments, it's not good:

```
cat /proc/dahdi/1
Span 1: WCTDM/0 "Wildcard TDM800P Board 1" (MASTER)
IRQ misses: 1762187
  1 WCTDM/0/0 FXOKS (In use)
  2 WCTDM/0/1 FXOKS (In use)
  3 WCTDM/0/2 FXOKS (In use)
  4 WCTDM/0/3 FXOKS (In use)
```

Digium gives some hints in their *Knowledge Base* here : http://kb.digium.com/entry/1/63/

PRI Digium cards needs 1000 interuption per seconds. If the systeme cannot supply them, it increment the IRQ missed counter.

As indicated in Digium *KB* you should avoid shared IRQ with other equipments (like HD or NIC interfaces).

Notes on configuration files

/etc/dahdi/system.conf

A *span* is created for each card port. Below is an example of a standard E1 port:

```
span=1,1,0,ccs,hdb3
dchan=16
bchan=1-15,17-31
echocanceller=mg2,1-15,17-31
```

Each span has to be declared with the following information:

```
span=<spannum>,<timing>,<LBO>,<framing>,<coding>[,crc4]
```

- `spannum` : corresponds to the span number. It starts to 1 and has to be incremented by 1 at each new span. This number MUST be unique.
- `timing` : describes the how this span will be considered regarding the synchronisation :
  - 0 : do not use this span as a synchronisation source,
  - 1 : use this span as the primary synchronisation source,
  - 2 : use this span as the secondary synchronisation source etc.
- `LBO` : 0 (not used)
- `framing` : correct values are `ccs` or `cas`. For ISDN lines, `ccs` is used.
- `coding` : correct valus are `hdb3` or `ami`. For example, `hdb3` is used for an E1 (PRI) link, whereas `ami` is used for T0 (french BRI) link.
- `crc4` : this is a framing option for PRI lines. For example it is rarely use in France.

Note that the `dahdi_genconf` command should usually give you the correct parameters (if you correctly set the cards jumper). All these information should be checked with your operator.

/etc/asterisk/chan_dahdi.conf

This file contains the general parameters of the DAHDI channel. It is not generated via the `dahdi_genconf` command.

/etc/asterisk/dahdi-channels.conf

This file contains the parameters of each channel. It is generated via the `dahdi_genconf` command.

1.8.17 Incall

General Configuration

You can configure incoming calls settings in *Services → IPBX → Call Management → Incoming calls*.

DID (Direct Inward Dialing) Configuration

When a "+" character is prepended a called DID, the "+" character is discarded.

Example:

> Bob has a DID with number 1000. Alice can call Bob by dialing either 1000 or +1000, without configuring another DID.

1.8.18 Interconnections

Interconnect two XiVO directly

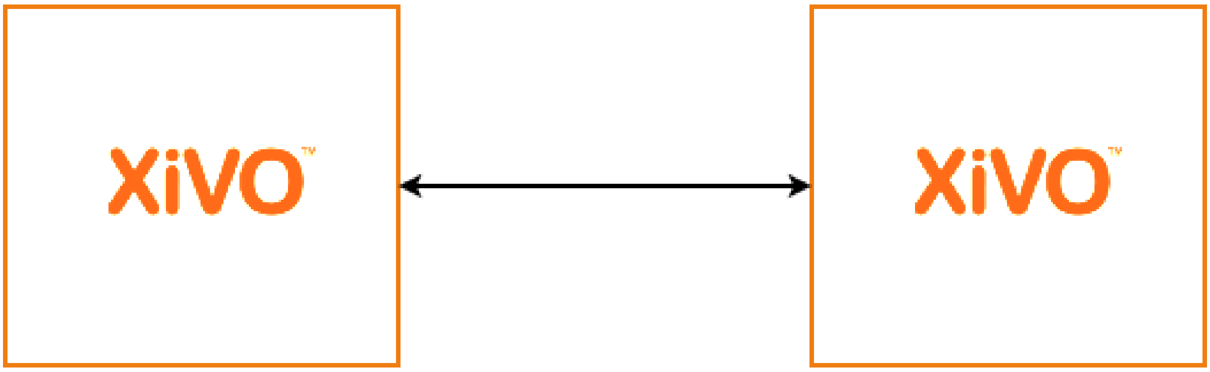


Figure 1.46: Situation diagram

Interconnecting two XiVO will allow you to send and receive calls between the users configured on both sides.

The steps to configure the interconnections are:

- Establish the trunk between the two XiVO, that is the SIP connection between the two servers
- Configure outgoing calls on the server(s) used to emit calls
- Configure incoming calls on the server(s) used to receive calls

For now, only SIP interconnections have been tested.

Establish the trunk

The settings below allow a trunk to be used in both directions, so it doesn't matter which server is A and which is B.

Consider XiVO A wants to establish a trunk with XiVO B.

On XiVO B, go on page *Services* → *IPBX* → *Trunk management* → *SIP Protocol*, and create a SIP trunk:

```
Name : xivo-trunk
Username: xivo-trunk
Password: pass
Connection type: Friend
IP addressing type: Dynamic
Context: <see below>
```

Note: For the moment, Name and Username need to be the same string.

The `Context` field will determine which extensions will be reachable by the other side of the trunk:

- If `Context` is set to `default`, then every user, group, conf room, queue, etc. that have an extension if the `default` context will be reachable directly by the other end of the trunk. This setting can ease configuration if you manage both ends of the trunk.
- If you are establishing a trunk with a provider, you probably don't want everything to be available to everyone else, so you can set the `Context` field to `Incalls`. By default, there is no extension available in this context, so we will be able to configure which extension are reachable by the other end. This is the role of the incoming calls: making bridges from the `Incalls` context to other contexts.

On XiVO A, create the other end of the SIP trunk on the *Services* → *IPBX* → *Trunk management* → *SIP Protocol*:

```
Name: xivo-trunk
Username: xivo-trunk
Password: pass
Identified by: Friend
Connection type: Static
Address: <XiVO B IP address or hostname>
Context: Incalls
```

Register tab:

```
Register: checked
Transport: udp
Username: xivo-trunk
Password: pass
Remote server: <XiVO B IP address or hostname>
```

On both XiVO, activate some codecs, *Services* → *IPBX* → *General Settings* → *SIP protocol*, tab `Signaling`:

```
Enabled codecs: at least GSM (audio)
```

At that point, the Asterisk command `sip show registry` on XiVO B should print a line showing that XiVO A is registered, meaning your trunk is established.

Set the outgoing calls

The outgoing calls configuration will allow XiVO to know which extensions will be called through the trunk.

On the call emitting server(s), go on the page *Services* → *IPBX* → *Call management* → *Outgoing calls* and add an outgoing call.

Tab General:

```
Trunks: xivo-trunk
```

Tab Exten:

```
Exten: **99. (note the period at the end)
Stripnum: 4
```

This will tell XiVO: if any extension begins with `**99`, then try to dial it on the trunk `xivo-trunk`, after removing the 4 first characters (the `**99` prefix).

The most useful special characters to match extensions are:

```
. (period): will match one or more characters
X: will match only one character
```

You can find more details about pattern matching in Asterisk (hence in XiVO) on the Asterisk wiki.

Set the incoming calls

Now that we have calls going out from a XiVO, we need to route incoming calls on the XiVO destination.

Note: This step is only necessary if the trunk is linked to an Incoming calls context.

To route an incoming call to the right destination in the right context, we will create an incoming call in *Services → IPBX → Call management → Incoming calls*.

Tab General:

```
DID: 101
Context: Incalls
Destination: User
Redirect to: someone
```

This will tell XiVO: if you receive an incoming call to the extension `101` in the context `Incalls`, then route it to the user `someone`. The destination context will be found automatically, depending on the context of the line of the given user.

So, with the outgoing call set earlier on XiVO A, and with the incoming call above set on XiVO B, a user on XiVO A will dial `**99101`, and the user `someone` will ring on XiVO B.

Interconnect a XiVO to a PBX via an ISDN link

The goal of this architecture can be one of:

- start a smooth migration between an old telephony system towards IP telephony with XiVO
- bring new features to the PBX like voicemail, conference, IVR etc.

First, XiVO is to be integrated transparently between the operator and the PBX. Then users or features are to be migrated from the PBX to the XiVO. It requires a special call routing configuration both on the XiVO and on the PBX.

Hardware

General uses You must have an ISDN card able to support both the ISDN provider and ISDN links with PBX.

Note: If you have two ISDN provider links to PBX, XiVO should have a card with 4 spans : two to the provider, and two to the PBX.

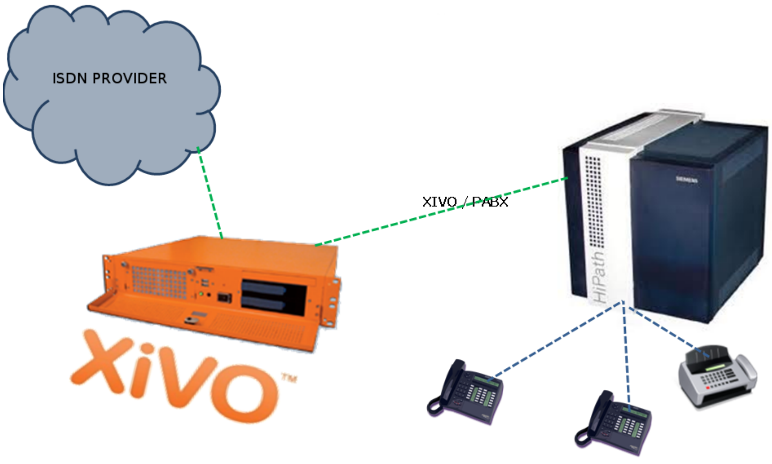


Figure 1.47: Interconnect a XiVO to a PBX

If you use two cards If you use two cards, you have to :

- Use a cable for clock synchronization between the cards
- Configure the "wheel" to define the cards order in the system. The ISDN links used by XiVO to synchronize have to be plugged on the card number one!

Please refer to the section *Sync cable*

Configuration

You have now to configure two files :

1. `/etc/dahdi/system.conf`
2. `/etc/asterisk/dahdi-channels.conf`

system.conf

Clock configuration

- Provider side - XiVO will get the clock from the provider : the `timing` value is to be different from 0 (see */etc/dahdi/system.conf* section)
- PBX side - XiVO will provide the clock to the PBX : the `timing` value is to be set to 0 (see */etc/dahdi/system.conf* section)

Example Below an example for interconnection with two ISDN provider:

```
# Span 1: TE4/0/1 "TE4XXP (PCI) Card 0 Span 1" (MASTER)
span=1,1,0,ccs,hdb3          # Span towards Provider
bchan=1-15,17-31
dchan=16
echocanceller=mg2,1-15,17-31

# Span 2: TE4/0/2 "TE4XXP (PCI) Card 0 Span 2"
span=2,2,0,ccs,hdb3          # Span towards Provider
```

```
bchan=32-46,48-62
dchan=47
echocanceller=mg2,32-46,48-62

# Span 3: TE4/0/3 "TE4XXP (PCI) Card 0 Span 3"
span=3,0,0,ccs,hdb3          # Span towards PBX
bchan=63-77,79-93
dchan=78
echocanceller=mg2,63-77,79-93

# Span 4: TE4/0/4 "TE4XXP (PCI) Card 0 Span 4"
span=4,0,0,ccs,hdb3          # Span towards PBX
bchan=94-108,110-124
dchan=109
echocanceller=mg2,94-108,110-124
```

dahdi-channels.conf

Configuraton Modify the file `/etc/asterisk/dahdi-channels.conf`

```
group : g0 provider side, g2 PBX side
context : from-extern or from-pabx
signalling : pri_cpe provider side, pri_net PBX side
```

> **Warning:** Towards certains destinations, some PBX use an overlapdialing (digits are sent one by one). In this case, we have to activate a parameter on the spans concerned:
>
> ```
> overlapdial = incoming
> ```

This can be seen with "pri intense debug"

Below an example of :file:`/etc/asterisk/dahdi-channels.conf`. Be careful to three parameters :

- group
- context
- signalling

Example

```
; Span 1: TE4/0/1 "TE4XXP (PCI) Card 0 Span 1" (MASTER)
group=0,11
context=from-extern
switchtype = euroisdn
signalling = pri_cpe
channel => 1-15,17-31

; Span 2: TE4/0/2 "TE4XXP (PCI) Card 0 Span 2"
group=0,12
context=from-extern
switchtype = euroisdn
signalling = pri_cpe
channel => 32-46,48-62

; Span 3: TE4/0/3 "TE2XXP (PCI) Card 0 Span 3"
group=2,13
context=from-pabx
overlapdial=incoming
switchtype = euroisdn
signalling = pri_net
```

```
channel => 63-77,79-93

; Span 4: TE4/0/4 "T4XXP (PCI) Card 0 Span 4"
group=2,14
context=from-pabx
overlapdial=incoming
switchtype = euroisdn
signalling = pri_net
channel => 94-108,110-124
```

Passthru function

Create the "from-pabx" context

- Create a file named xxxxx.conf (where xxxxx is the customer name) in the directory `/etc/asterisk/extensions_extra.d/`.
- Add the following lines in the file:

```
[from-pabx]
exten = _X.,1,NoOp(« Appel depuis Pabx »)
exten = _X.,n,goto(default,${EXTEN},1)
```

This dialplan allows to route incoming calls from the PBX in the default context of XiVO. Then, calls are routed :

- Or to a SIP phone (in default context)
- Or to the outgoing (to-extern context included in default context)

Create the "to-extern" context In the webi, create a context named "to-pabx" :

```
Name : to-pabx
Display Name : to-pabx
Context type : Outcall
Include sub-contexts : No context inclusion
```

This context allows to route incoming calls from the XiVO to the PBX.

Create incoming calls In our example, incoming calls on spans 1 and 3 (spans pluged to the provider) are routed by from-extern context. We are going to create a default route to redirect incoming calls to the PBX.

Create an incoming call as below :

```
DID : XXXX (according to the number of digits sent by the provider)
Context : Incoming calls
Destination : Customized
Command : Goto(to-pabx,${XIVO_DSTNUM},1)
```

Create the interconnections You have to create two interconnections :

- provider side : dahdi/g0
- PBX side : dahdi/g2

In the menu *Services* → *IPBX* → *Trunk management* → *Customized* page

```
Name : t2-operateur
Interface : dahdi/g0
Contexte : to-extern
```

The second interconnection

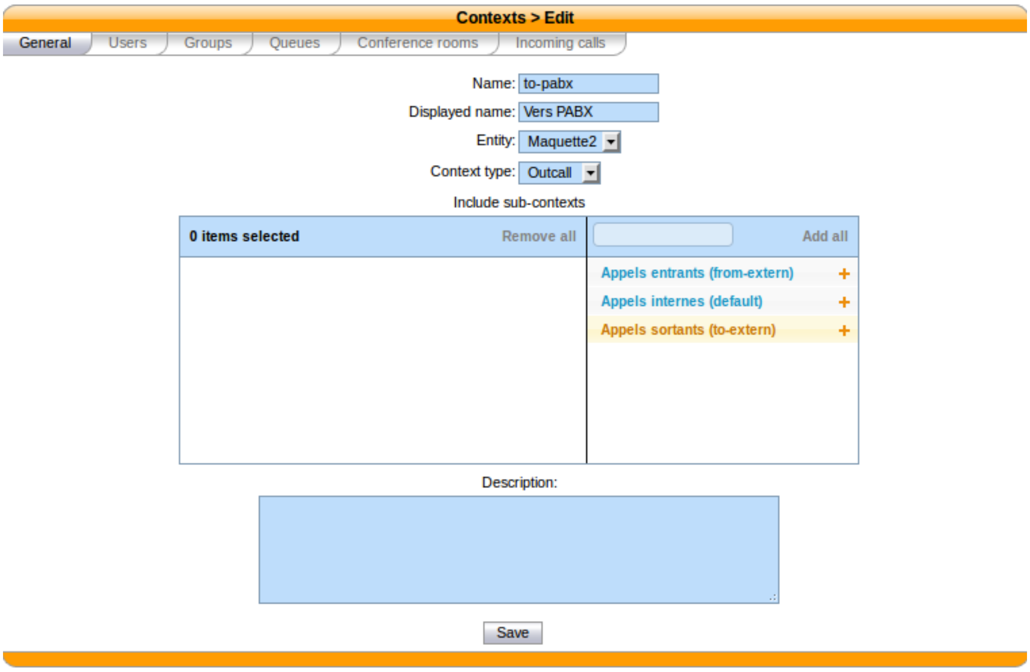


Figure 1.48: to-extern context

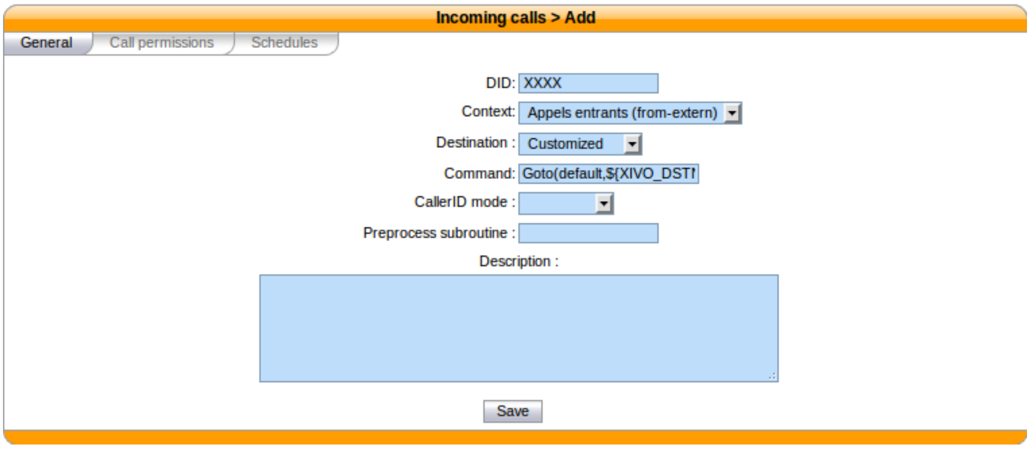


Figure 1.49: Incoming call XXXX

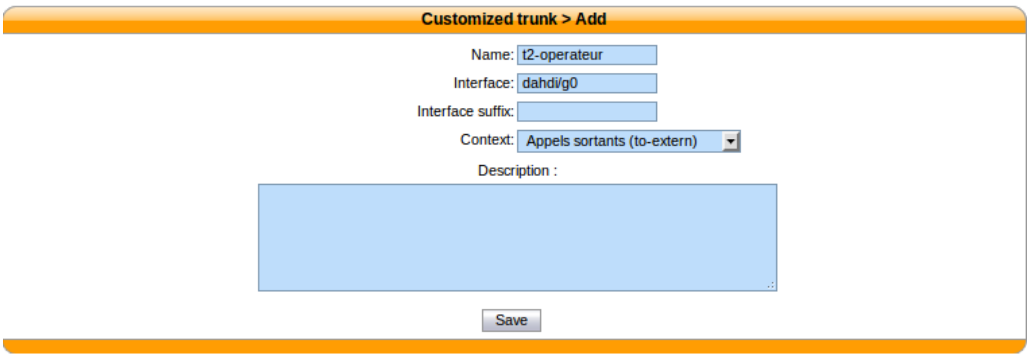


Figure 1.50: Customized interconnection

```
Name : t2-pabx
Interface : dahdi/g2
Context : to-pabx
```

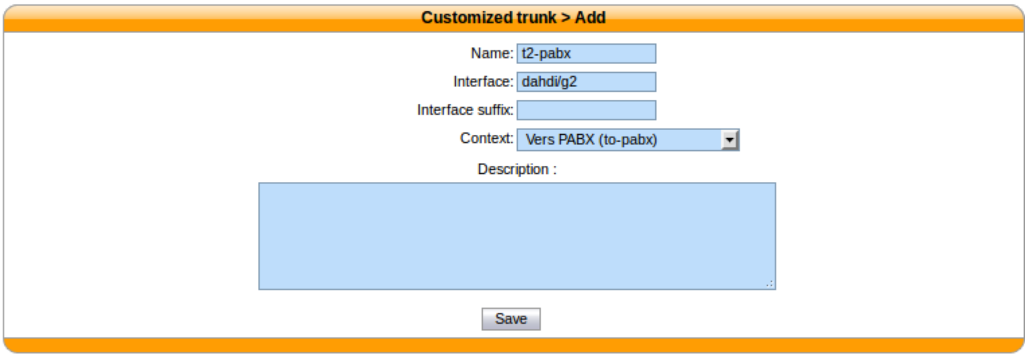


Figure 1.51: Customized interconnection

Create outgoing calls You must create two rules of outgoing calls in the menu *Services* → *IPBX* → *Call management* → *Outgoing calls* page

1. Redirect calls to the PBX

```
Name : fsc-pabx
Context : to-pabx
Trunks : choose the "t2-pabx" interconnection
```

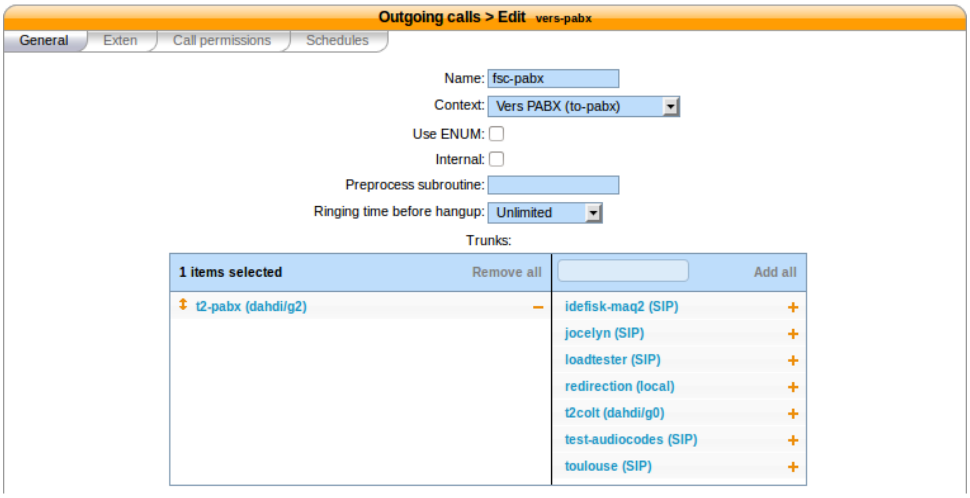


Figure 1.52: Outgoing call

In the extensions tab

```
exten : XXXX
```

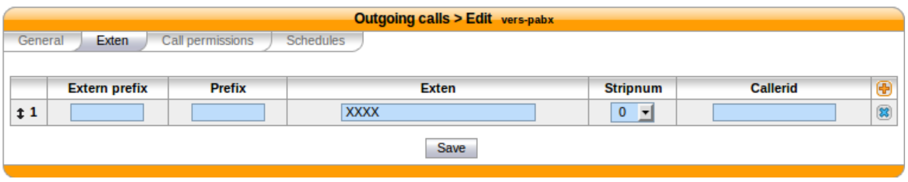


Figure 1.53: Outgoing call

2. Rename the rule "default" in "fsc-operateur":

   ```
   Name : fsc-operateur
   Context : to-extern
   Trunks : choose the "t2-operateur" interconnection
   ```

In the extensions tab:

```
exten = X.
```

Create an interconnection

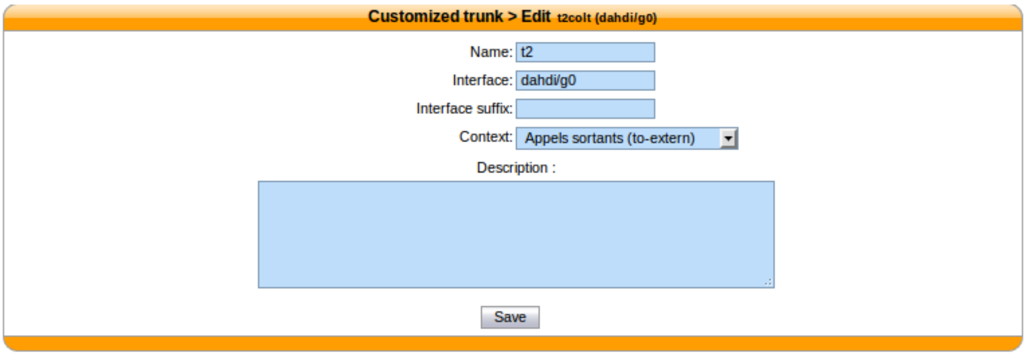
There are three types of interconnections :

- Customized
- SIP
- IAX

Customized interconnection

Add an interconnection to the menu *Services* → *IPBX* → *Trunk management* → *Customized*

```
Name : interconnection name
Interface : dahdi/g0
Context : outgoing call (to-extern)
```



Debug

Interesting Asterisk commands:

```
sip show peers
sip show registry
sip set debug on
```

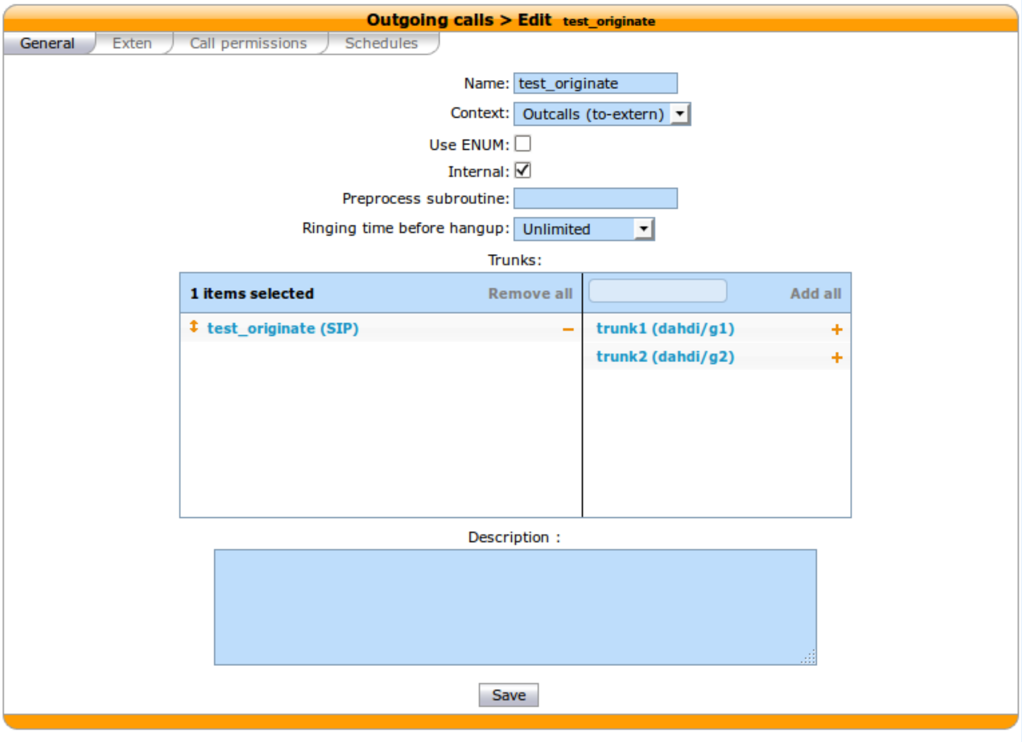
Caller ID

When setting up an interconnection with the public network or another PBX, it is possible to set a caller ID in different places. Each way to configure a caller ID has it's own use case.

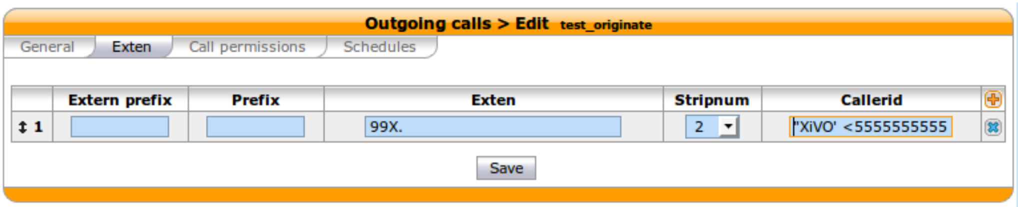
The format for a caller ID is the following `"My Name" <9999>` If you don't set the number part of the caller ID, the dialplan's number will be used instead. This might not be a good option in most cases.

Outgoing call caller ID

When you create an outgoing call, it's possible to set the it to internal, using the check box in the outgoing call configuration menu. When this option is activated, the caller's caller ID will be forwarded to the trunk. This option is use full when the other side of the trunk can reach the user with it's caller ID number.



When the caller's caller ID is not usable to the called party, the outgoing call's caller id can be fixed to a given value that is more use full to the outside world. Giving the public number here might be a good idea.



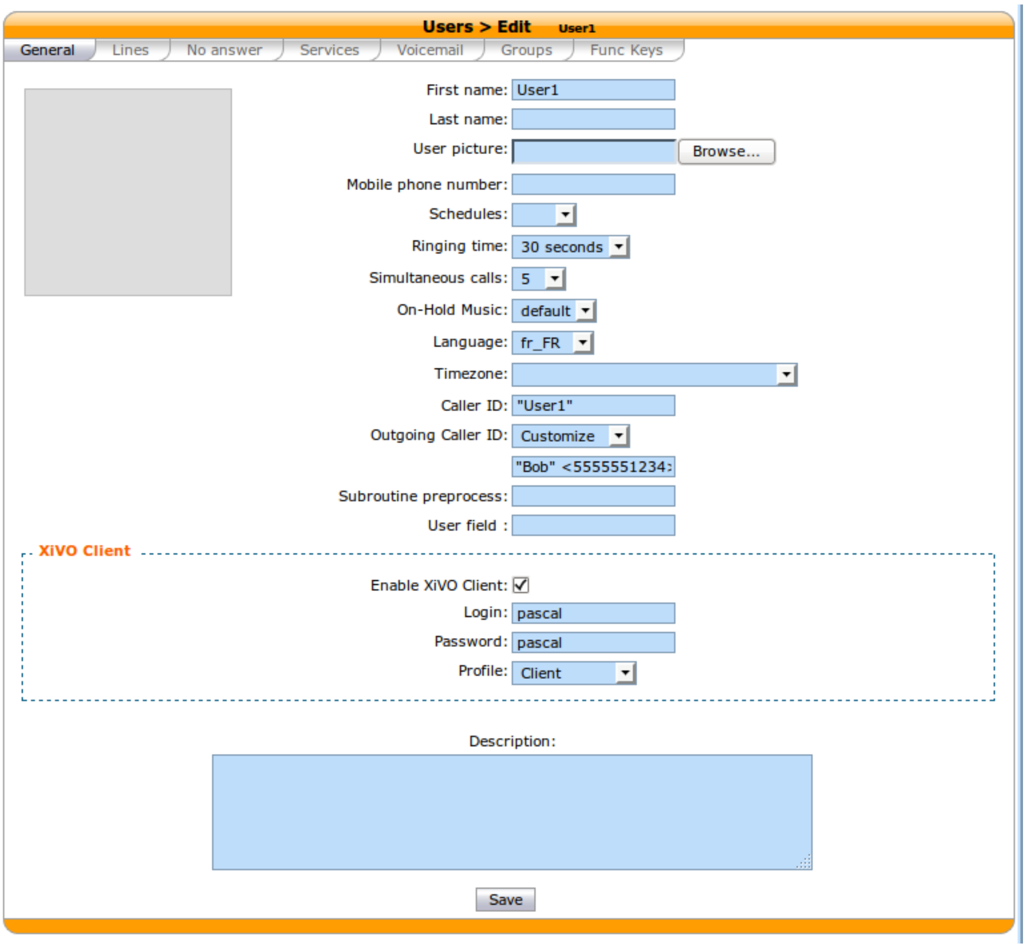
A user can also have a forced caller ID for outgoing calls. This can be use full for someone who has his own public number. This option can be set in the user's configuration page. The Outgoing Caller ID id option must be set to Customize. The user can also set his outgoing caller ID to anonymous.

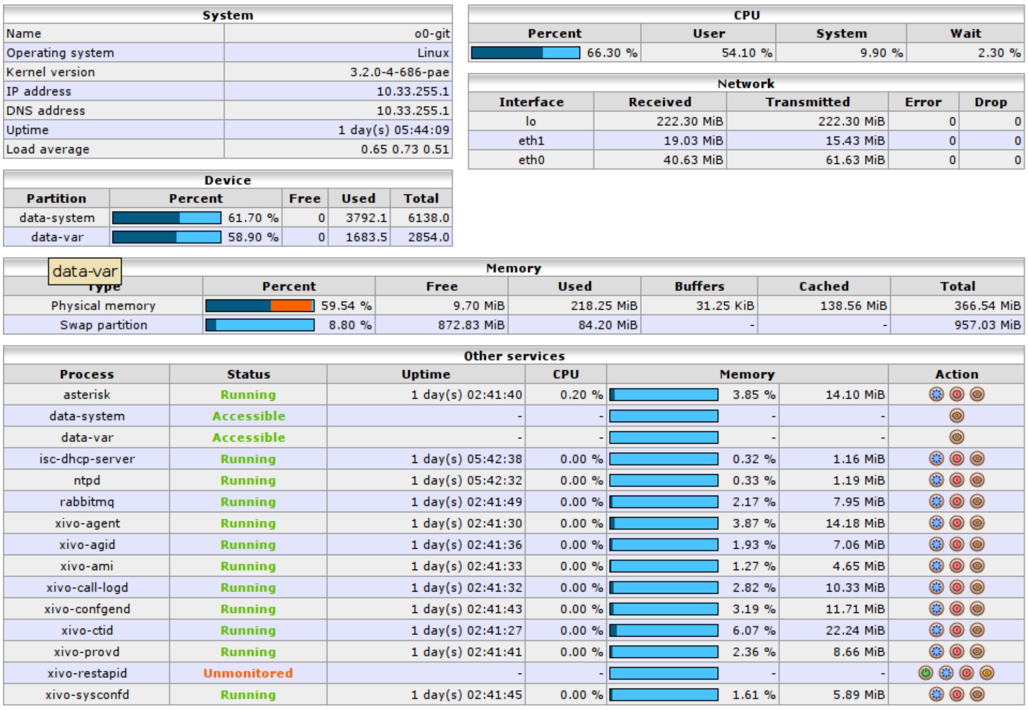
The order of precedence when setting the caller ID in multiple place is the following.

1. Internal
2. User's outgoing caller ID
3. Outgoing call
4. Default caller ID

1.8.19 Monitoring

The Monitoring section gives an overview of a XiVO system's status and of all monitored processes. It is divided into 6 sections :


Users > Edit User1
General
Lines
No answer
Services
Voicemail
Groups
Func Keys
First name: User1
Last name:
User picture:
Browse...
Mobile phone number:
Schedules:
Ringing time: 30 seconds
Simultaneous calls: 5
On-Hold Music: default
Language: fr_FR
Timezone:
Caller ID: "User1"
Outgoing Caller ID: Customize
"Bob" <5555551234:
Subroutine preprocess:
User field :
XiVO Client
Enable XiVO Client:
Login: pascal
Password: pascal
Profile: Client
Description:
Save



System

Displays generic information about the operating system, network adresses, uptime and load average. Read only.

Device

Displays free/used space on physical storage partitions. Read only.

CPU

Monitors the CPU usage. Read only.

Network

Displays network interfaces and corresponding network traffic. Read only.

Memory

Displays Physical and swap memory usage. Read only.

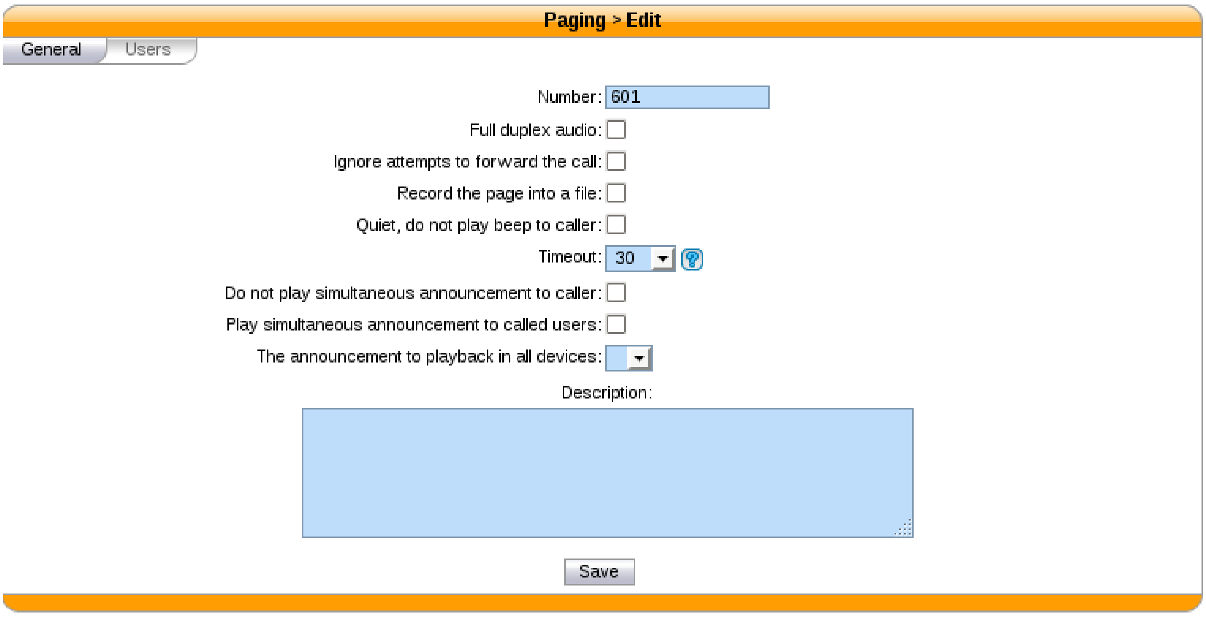
Other Services

Lists XiVO related processes (most of which are daemons) with their corresponding status, uptime, ressource usage and controls to Restart service (blue button), stop service (red button) and stop monitoring service (grey button).

1.8.20 Paging

With XiVO, you can define paging (i.e. intercom) extensions to page a group of users. When calling a paging extension, the phones of the specified users will auto-answer, if they support it.

You can manage your paging extensions via the *Services* → *IPBX* → *Paging* page.



When adding a new paging extension, the number can be any numeric value; to call it, you just need to prefix the paging number with `*11`.

1.8.21 Parking

With XiVO it is possible to park calls, the same way you may park your car in a car parking. If you define supervised keys on a phone set for all the users of a system, when a call is parked, all the users are able to see that some one is waiting for an answer, push the phone key and get the call back to the phone.

There is a default parking number, 700, which is already configured when you install XiVO, but you may change the default configuration by editing the parking extension in menu *Service* → *IPBX* → *IPBX Services* → *Extensions* → *Advanced* → *Parking*

Using this extension, you may define the parking number used to park call, the parking lots, wether the sytem is rotating over the parking lots to park the calls, enable parking hint if you want to be able to supervise the parking using phone keys and other system default parameters.

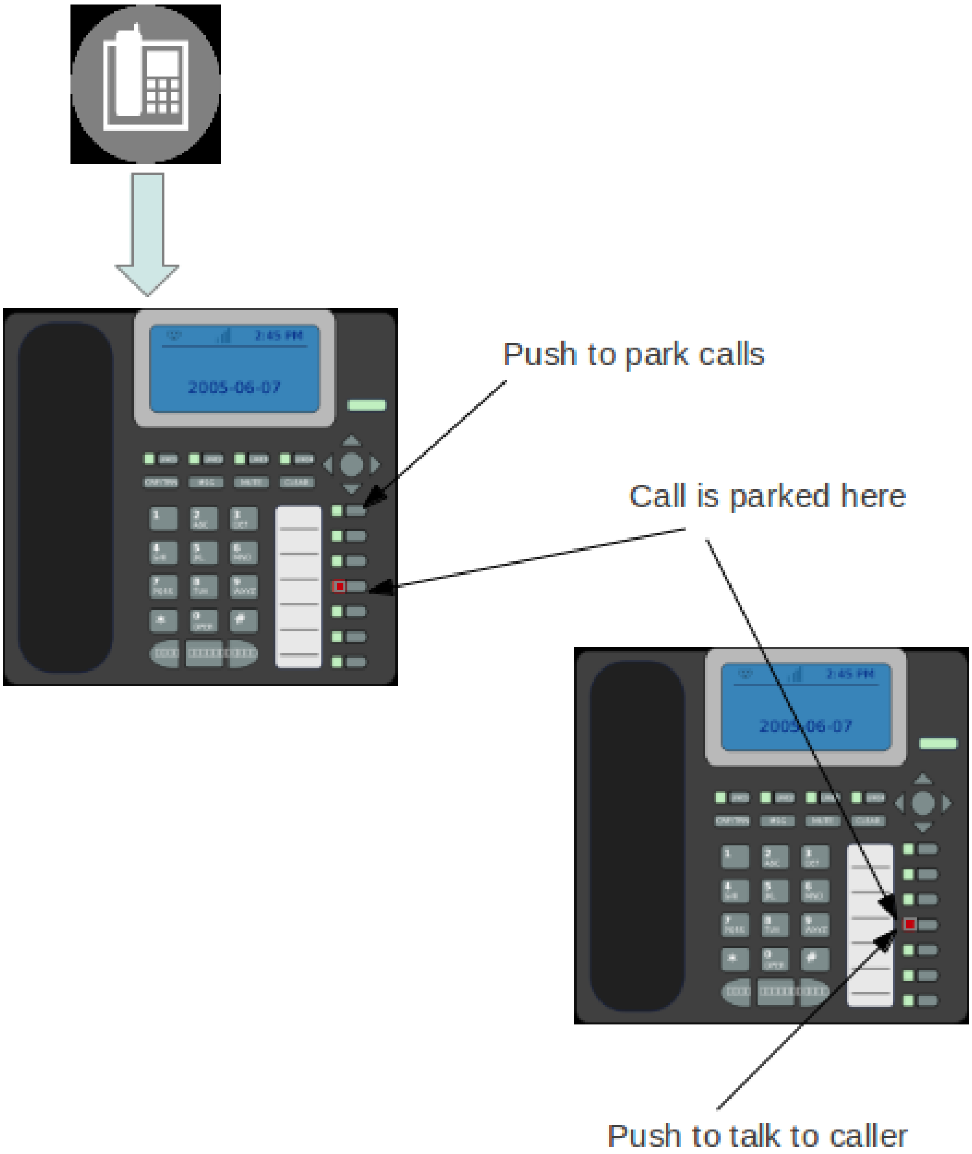
You have two options in case of parking timeout :

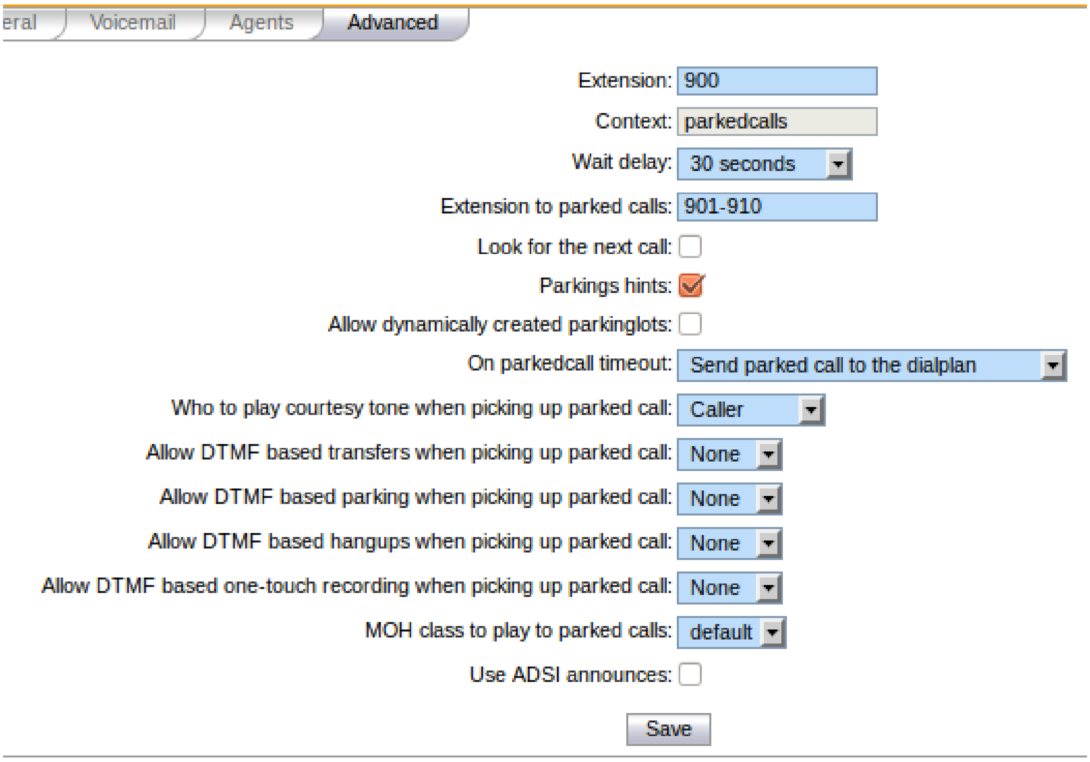
- Callback the peer that parked this call

  In this case the call is sent back to the user who parked the call.

- Send park call to the dialplan

  In case you don't want to call back the user who parked the call, you have the option to send the call to any other extension or application. If the parking times out, the call is sent back to the dialplan in context


Push to park calls
Call is parked here
Push to talk to caller

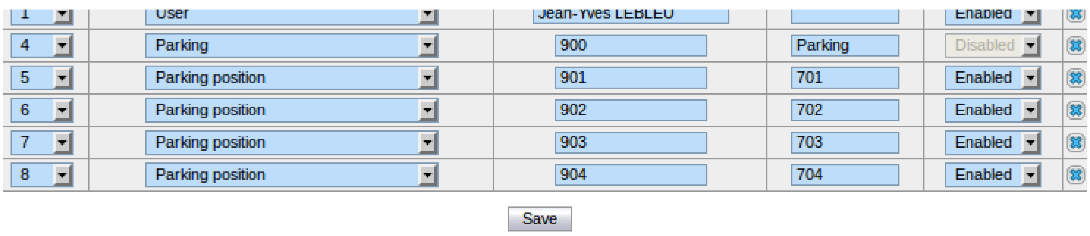


[parkedcallstimeout]. You can define this context in a dialplan configuration file *Service* → *IPBX* → *Configuration Files* where you may define this context with dialplan commands.

Example:

```
[parkedcallstimeout]
exten = s,1,Noop('park call time out')
same  =   n,Playback(hello-world)
same  =   n,Hangup()
```

It is also usual to define supervised phone keys to be able to park and unpark calls as in the example below.



1.8.22 Phonebook

A global phone book can be defined in *IPBX Service -> Phone book*. The phone book can be used from the XiVO client, from the phones directory look key if the phone is compatible and are used to set the Caller ID for incoming calls.

You can add entries one by one or you can mass-import from a CSV file.

Mass-import contacts

Go in the *IPBX Services -> Phonebook* section and move your mouse cursor on the + button in the upper right corner. Select *Import a file*.

The file to be imported must be a CSV file, with a pipe character | as field delimiter. The file must be encoded in UTF-8.

Mandatory headers are :

- title (possible values : "mr", "mrs", "ms")
- displayname

Optional headers are :

- firstname
- lastname
- society
- mobilenumber [9]
- email
- url
- description
- officenumber [1]
- faxnumber [1]
- officeaddress1
- officeaddress2
- officecity
- officestate
- officezipcode
- officecountry [10]
- homenumber [1]
- homeaddress1
- homeaddress2
- homecity
- homestate
- homezipcode
- homecountry [2]
- othernumber [1]
- otheraddress1
- otheraddress2
- othercity
- otherstate
- otherzipcode
- othercountry [2]

[9] These fields must contain only numeric characters, no space, point, etc.
[10] These fields must contain ISO country codes. The complete list is described here.

Displayed fields

It's possible to add more fields to the display in the CTI client. The display can be customized in the web interface under *Services -> CTI server -> Directories -> Display filter*.

Fields that can be displayed are set in *Directories -> Definitions -> xivodir*

The field name will be used to refer to this field in the directory display.

The fields in definition can be used with the following syntax *{db-[field-name]}*

General phone book section

These fields are set in the General tab of the phone book.

- phonebook.description
- phonebook.displayname
- phonebook.email
- phonebook.firstname
- phonebook.fullname (this value is automatically generated as *"<firstname> <lastname>"*, e.g. *"John Doe"*)
- phonebook.lastname
- phonebook.society
- phonebook.title
- phonebook.url

Phone numbers

These are the different phone numbers that are available

- phonebooknumber.fax
- phonebooknumber.home
- phonebooknumber.mobile
- phonebooknumber.office
- phonebooknumber.other

Addresses

Each configured address can be accessed

Address uses the following syntax *phonebookaddress.[location].[field]*, e.g. *phonebookaddress.office.zipcode*.

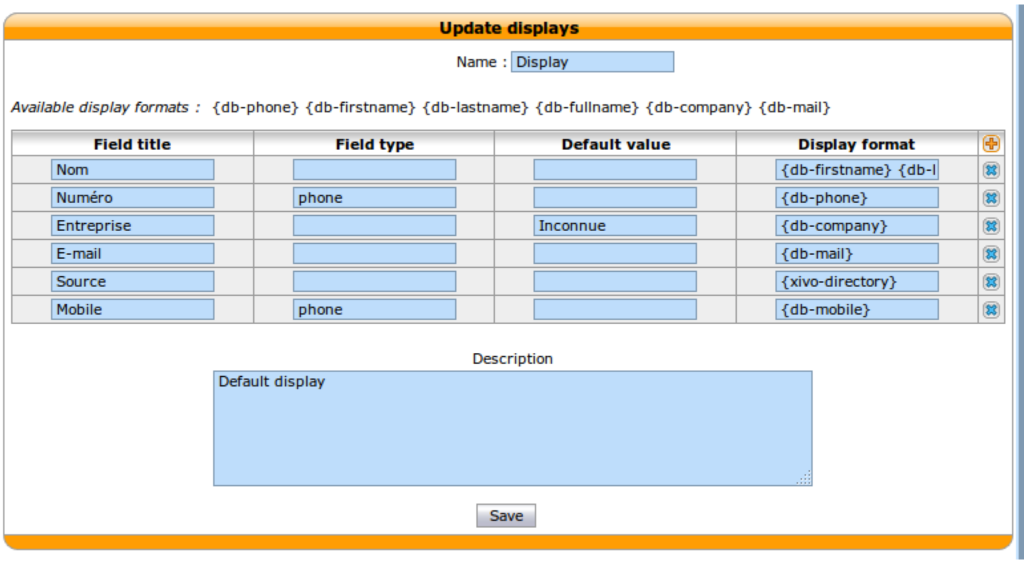
Locations

- home
- office
- other

Fields

- address1
- address2
- city
- country
- state
- zipcode

Each line is a field that will be displayed in the Remote Directory xlet.



Adding the fax to the directory display

1. In the definition section, add a field name *fax* with the value *phonebooknumber.fax.number*.
2. In the display filter section add a field with field title *Fax* and display format *{db-fax}*.
3. Restart the CTI Server

Now the fax should be available displayed in the Remote Directory xlet.

Reverse lookup

It's possible to do reverse lookups on incoming calls to show a better caller ID name when the caller is in one of our directories.

Reverse lookup will only be tried if at least one of the following conditions is true:

- The caller ID name is the same as the caller ID number
- The caller ID name is "unknown"

Also, reverse lookup is performed after *caller ID number normalization* (since XiVO 13.11).

Some configuration must be in place to enable reverse directory lookups.

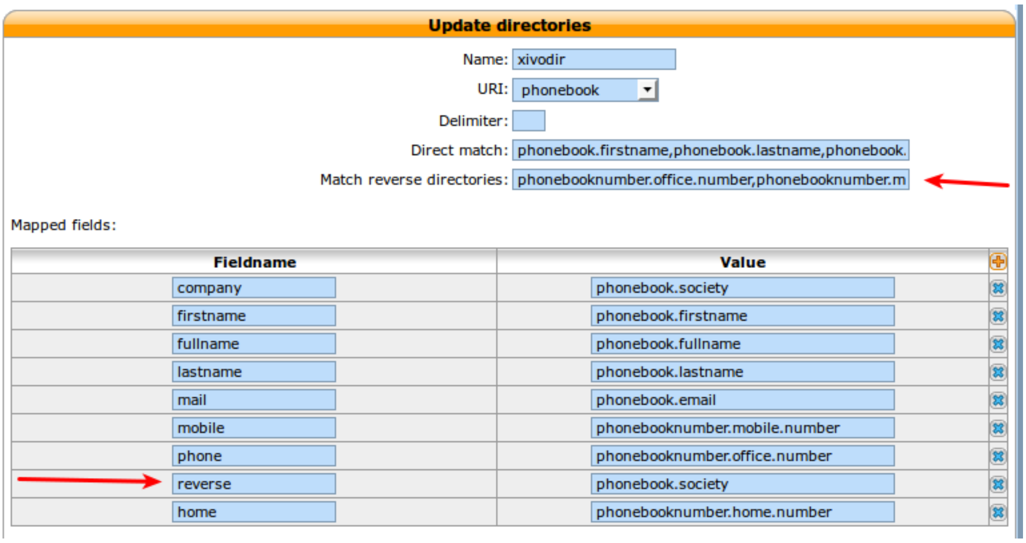
Match reverse fields

The *Match reverse directories* field in *Services* → *CTI Server* → *Directories* → *Definitions* should contains the fields that are used by the reverse lookup. The list is comma separated and each field that appears in this list must also appear in the *Value* column of the *Mapped Fields* section, or the reverse lookup won't work.

Example:

```
phonebooknumber.office.number,phonebooknumber.mobile.number,phonebooknumber.home.number
```

This line would match office, home and mobile numbers on incoming calls.



Displayed field

You have now to define which field should be used to display the result. This is done by defining a mapping between the field named *reverse* on the *Mapped fields* list and a database field.

Examples:

```
reverse => phonebook.society
or
reverse => phonebook.fullname
```

The first example would show the contact's company name on the caller ID name, the second would show his full name.

Include the directory

To include a directory in reverse directory definition go to *Services* → *CTI Server* → *Directories* → *Reverse directories* and add the directories to include to reverse lookups in the *Related directories* section.

Restart the CTI server and incoming caller IDs should be resolved using the specified directories.

1.8.23 Provisioning

XiVO supports the auto-provisioning of a large number of telephony *Devices*, including SIP phones, SIP ATAs, and even softphones.

Introduction

The auto-provisioning feature found in XiVO make it possible to provision, i.e. configure, a lots of telephony devices in an efficient and effortless way.

How it works

Here's a simplified view of how auto-provisioning is supported on a typical SIP hardphone:

1. The phone is powered on
2. During its boot process, the phone sends a DHCP request to obtain its network configuration
3. A DHCP server replies with the phone network configuration + an HTTP URL
4. The phone use the provided URL to retrieve a common configuration file, a MAC-specific configuration file, a firmware image and some language files.

Building on this, configuring one of the supported phone on XiVO is as simple as:

1. *Configuring the DHCP Server*
2. *Installing the required provd plugin*
3. Powering on the phone
4. Dialing the user's provisioning code from the phone

And *voila*, once the phone has rebooted, your user is ready to make and receive calls. No manual editing of configuration files nor fiddling in the phone's web interface.

Limitations

- Device synchronisation does not work in the situation where multiple devices are connected from behind a NAPT network equipment. The devices must be resynchronised manually.

External links

- Introduction to provd plugin model
- HTTP/TFTP requests processing in provd - part 1
- HTTP/TFTP requests processing in provd - part 2

Basic Configuration

You have two options to get your phone to be provisioned:

- Set up a DHCP server
- Tell manually each phone where to get the provisioning informations

You may want to manually configure the phones if you are only trying XiVO or if your network configuration does not allow the phones to access the XiVO DHCP server.

You may want to set up a DHCP server if you have a significant number of phones to connect, as no manual intervention will be required on each phone.

Configuring the DHCP Server

XiVO includes a DHCP server that facilitate the auto-provisioning of telephony devices. It is *not* activated by default.

There's a few things to know about the peculiarities of the included DHCP server:

- it only answers to DHCP requests from *supported devices*.
- it only answers to DHCP requests coming from the VoIP subnet (see *network configuration*).

This means that if your phones are on the same broadcast domain than your computers, and you would like the DHCP server on your XiVO to handle both your phones and your computers, that won't do it.

The DHCP server is configured via the *Configuration → Network → DHCP* page:

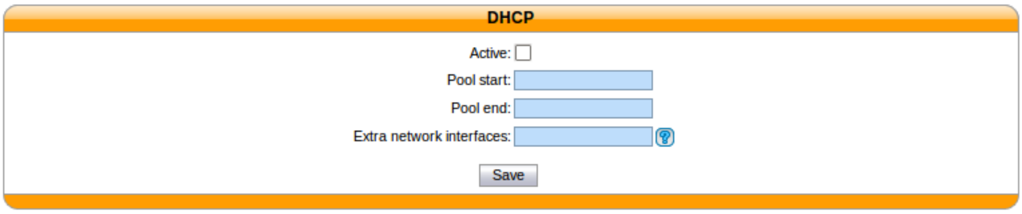


Figure 1.54: *Configuration → Network → DHCP*

Active Activate/desactivate the DHCP server.

Pool start The lower IP address which will be assigned dynamically. This address should be in the VoIP subnet. Example: `10.0.0.10`.

Pool end The higher IP address which will be assigned dynamically. This address should be in the VoIP subnet. Example: `10.0.0.99`.

Extra network interfaces A list of space-separated network interface name. Example: `eth0`.

Useful if you have done some custom configuration in the `/etc/dhcp/dhcpd_extra.conf` file. You need to explicitly specify the additional interfaces the DHCP server should listen on.

After saving your modifications, you need to click on *Apply system configuration* for them to be applied.

Installing `provd` Plugins

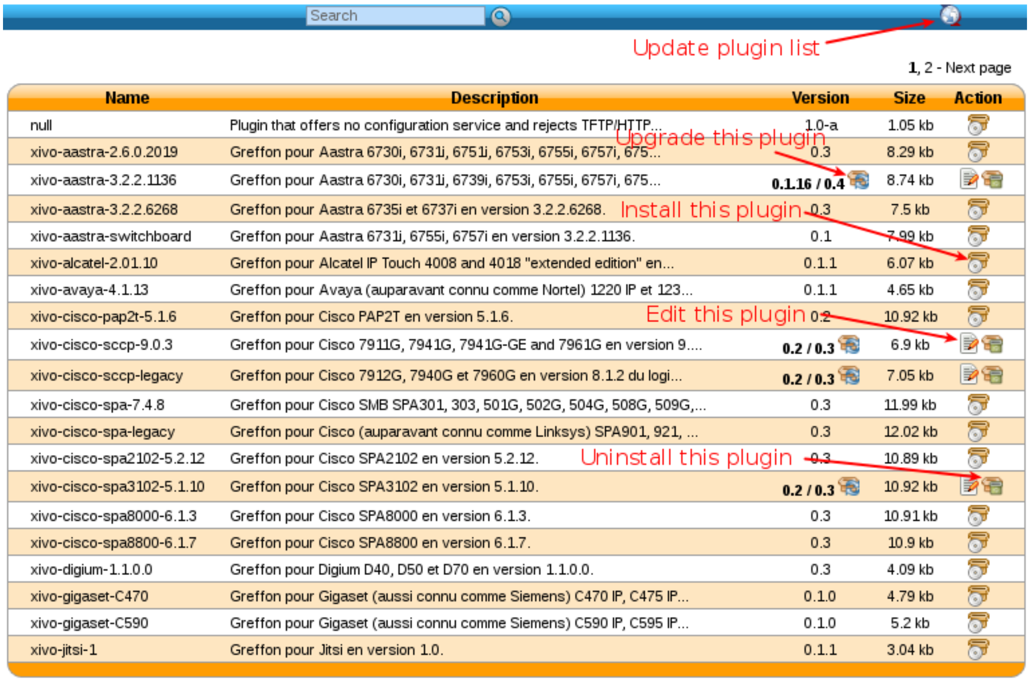
The installation and management of `provd` plugins is done via the *Configuration → Provisioning → Plugin* page:

The page shows the list of both the installed and installable plugins. You can see if a plugin is installed or not by looking at the *Action* column.

Here's the list of other things that can be done from this page:

- update the list of installable plugins, by clicking on the top right icon. On a fresh XiVO installation, this is the first thing to do.
- install a new plugin
- upgrade an installed plugin
- uninstall an installed plugin
- edit an installed plugin, i.e. install/uninstall optional files that are specific to each plugin, like firmware or language files

After installing a new plugin, you are automatically redirected to its edit page. You can then download and install optional files specific to the plugin. You are strongly advised to install firmware and language files for the phones you'll use although it's often not a strict requirement for the phones to work correctly.



Name	Description	Version	Size	Action
null	Plugin that offers no configuration service and rejects TFTP/HTTP...	1.0-a	1.05 kb	
xivo-aastra-2.6.0.2019	Greffon pour Aastra 6730i, 6731i, 6751i, 6753i, 6755i, 6757i, 675...	0.3	8.29 kb	
xivo-aastra-3.2.2.1136	Greffon pour Aastra 6730i, 6731i, 6739i, 6753i, 6755i, 6757i, 675...	0.1.16 / 0.4	8.74 kb	
xivo-aastra-3.2.2.6268	Greffon pour Aastra 6735i et 6737i en version 3.2.2.6268.	0.3	7.5 kb	
xivo-aastra-switchboard	Greffon pour Aastra 6731i, 6755i, 6757i en version 3.2.2.1136.	0.1	7.99 kb	
xivo-alcatel-2.01.10	Greffon pour Alcatel IP Touch 4008 and 4018 "extended edition" en...	0.1.1	6.07 kb	
xivo-avaya-4.1.13	Greffon pour Avaya (auparavant connu comme Nortel) 1220 IP et 123...	0.1.1	4.65 kb	
xivo-cisco-pap2t-5.1.6	Greffon pour Cisco PAP2T en version 5.1.6.	0.2	10.92 kb	
xivo-cisco-sccp-9.0.3	Greffon pour Cisco 7911G, 7941G, 7941G-GE and 7961G en version 9....	0.2 / 0.3	6.9 kb	
xivo-cisco-sccp-legacy	Greffon pour Cisco 7912G, 7940G et 7960G en version 8.1.2 du logi...	0.2 / 0.3	7.05 kb	
xivo-cisco-spa-7.4.8	Greffon pour Cisco SMB SPA301, 303, 501G, 502G, 504G, 508G, 509G,...	0.3	11.99 kb	
xivo-cisco-spa-legacy	Greffon pour Cisco (auparavant connu comme Linksys) SPA901, 921, ...	0.3	12.02 kb	
xivo-cisco-spa2102-5.2.12	Greffon pour Cisco SPA2102 en version 5.2.12.	0.3	10.89 kb	
xivo-cisco-spa3102-5.1.10	Greffon pour Cisco SPA3102 en version 5.1.10.	0.2 / 0.3	10.92 kb	
xivo-cisco-spa8000-6.1.3	Greffon pour Cisco SPA8000 en version 6.1.3.	0.3	10.91 kb	
xivo-cisco-spa8800-6.1.7	Greffon pour Cisco SPA8800 en version 6.1.7.	0.3	10.9 kb	
xivo-digium-1.1.0.0	Greffon pour Digium D40, D50 et D70 en version 1.1.0.0.	0.3	4.09 kb	
xivo-gigaset-C470	Greffon pour Gigaset (aussi connu comme Siemens) C470 IP, C475 IP...	0.1.0	4.79 kb	
xivo-gigaset-C590	Greffon pour Gigaset (aussi connu comme Siemens) C590 IP, C595 IP...	0.1.0	5.2 kb	
xivo-jitsi-1	Greffon pour Jitsi en version 1.0.	0.1.1	3.04 kb	

Figure 1.55: *Configuration* → *Provisioning* → *Plugin*

Warning: If you uninstall a plugin that is used by some of your devices, they will be left in an unconfigured state and won't be associated to another plugin automatically.

The search box at the top comes in handy when you want to find which plugin to install for your device. For example, if you have a Cisco SPA508G, enter "508" in the search box and you should see there's 1 plugin compatible with it.

Note: If your device has a number in its model name, you should use only the number as the search keyword since this is what usually gives the best results.

It's possible there will be more than 1 plugin compatible with a given device. In these cases, the difference between the two plugins is usually just the firmware version the plugins target. If you are unsure about which version you should install, you should look for more information on the vendor website.

It's good practice to only install the plugins you need and no more.

Alternative plugins repository By default, the list of plugins available for installation are the stable plugins for the officially supported devices.

This can be changed in the *Configuration* → *Provisioning* → *General* page, by setting the *URL* field to one of the following value:

- `http://provd.xivo.fr/plugins/1/stable/` – *officially supported devices* "stable" repository (*default*)
- `http://provd.xivo.fr/plugins/1/testing/` – officially supported devices "testing" repository
- `http://provd.xivo.fr/plugins/1/archive/` – officially supported devices "archive" repository
- `http://provd.xivo.fr/plugins/1/addons/stable/` – *community supported devices* "stable" repository

- `http://provd.xivo.fr/plugins/1/addons/testing/` – community supported devices "testing" repository

The difference between the stable and testing repositories is that the latter might contain plugins that are not working properly or are still in developement.

The archive repository contains plugins that were once in the stable repository.

After setting a new URL, you must refresh the list of installable plugins by clicking the update icon of the *Configuration → Provisioning → Plugin* page.

How to manually tell the phones to get their configuration

If you have set up a DHCP server on XiVO and the phones can access it, you can skip this section.

The according provisioning plugins must be installed.

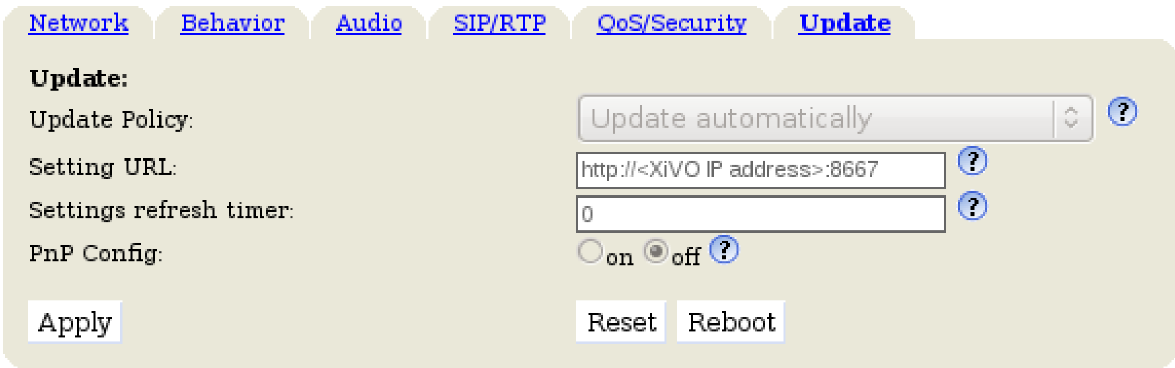
Aastra On the web interface of your phone, go to *Advanced settings → Configuration server*, and enter the following settings:

Polycom On the phone, go to *Menu → Settings → Advanced → Admin Settings → Network configuration → Server Menu* and enter the following settings:

- Server type: HTTP
- Server address: `http://<XiVO IP address>:8667/000000000000.cfg`

Then save and reboot the phone.

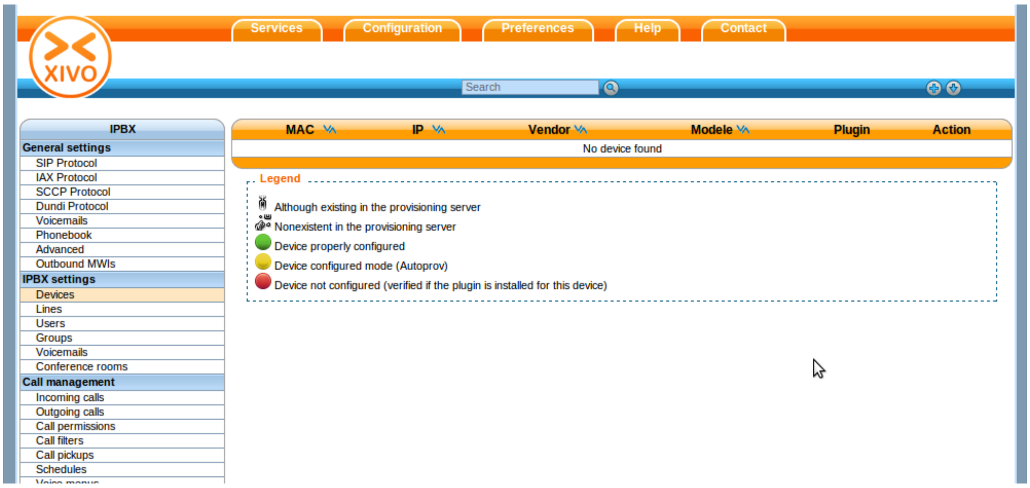
Snom On the web interface of your phone, go to *Setup → Advanced → Update* and enter the following settings:



Autoprovisioning a Device

Once you have installed the proper provd plugins for your devices and setup correctly your DHCP server, you can then connect your devices to your network.

But first, go to *Services* → *IPBX* → *Devices* page. You will then see that no devices are currently known by your XiVO:



You can then power on your devices on your LAN. For example, after you power on an Aastra 6731i and give it the time to boot and maybe upgrade its firmware, you should then see the phone having its first line configured as 'autoprov', and if you refresh the devices page, you should see that your XiVO now knows about your 6731i:

You can then dial from your Aastra 6731i the provisioning code associated to a line of one of your user. You will hear a prompt thanking you and your device should then reboot in the next few seconds. Once the device has rebooted, it will then be properly configured for your user to use it. And also, if you update the device page, you'll see that the icon next to your device has now passed to green:

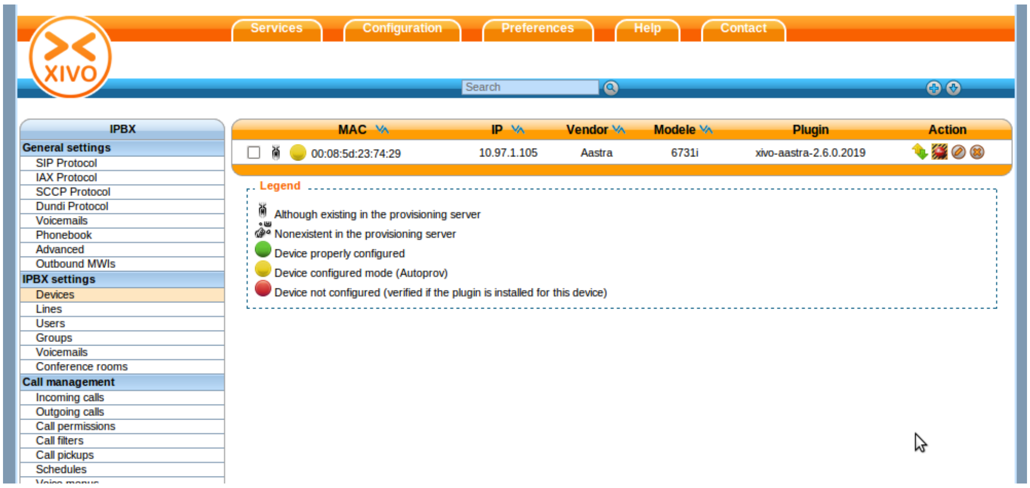
Resetting a Device

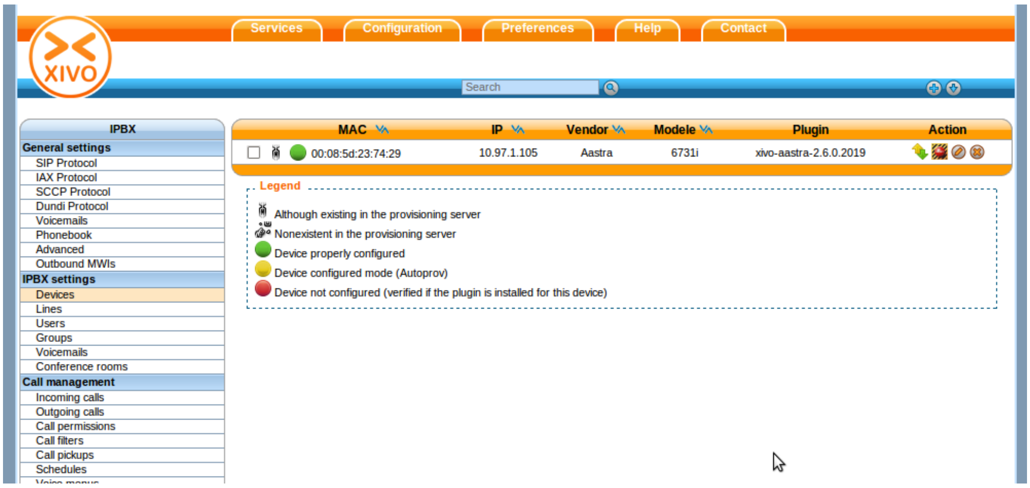
From the Device List in the Webi To remove a phone from XiVO or enable a device to be used for another user there are two different possibilities :

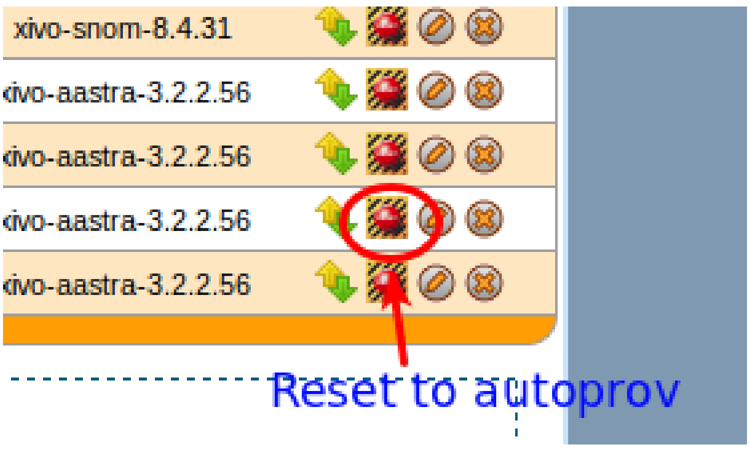
- click on the `reset to autoprov` button on the web interface

The phone will restarts and display autoprov, ready to be used for another user.

From the User Form in the Webi


Services
Configuration
Preferences
Help
Contact
XIVO
Search
IPBX
General settings
SIP Protocol
IAX Protocol
SCCP Protocol
Dundi Protocol
Voicemails
Phonebook
Advanced
Outbound MWIs
IPBX settings
Devices
Lines
Users
Groups
Voicemails
Conference rooms
Call management
Incoming calls
Outgoing calls
Call permissions
Call filters
Call pickups
Schedules
MAC
IP
Vendor
Modele
Plugin
Action
00:08:5d:23:74:29
10.97.1.105
Aastra
6731i
xivo-aastra-2.6.0.2019
Legend
Although existing in the provisioning server
Nonexistent in the provisioning server
Device properly configured
Device configured mode (Autoprov)
Device not configured (verified if the plugin is installed for this device)


Services
Configuration
Preferences
Help
Contact
XIVO
Search
IPBX
General settings
SIP Protocol
IAX Protocol
SCCP Protocol
Dundi Protocol
Voicemails
Phonebook
Advanced
Outbound MWIs
IPBX settings
Devices
Lines
Users
Groups
Voicemails
Conference rooms
Call management
Incoming calls
Outgoing calls
Call permissions
Call filters
Call pickups
Schedules
MAC
IP
Vendor
Modele
Plugin
Action
00:08:5d:23:74:29
10.97.1.105
Aastra
6731i
xivo-aastra-2.6.0.2019
Legend
Although existing in the provisioning server
Nonexistent in the provisioning server
Device properly configured
Device configured mode (Autoprov)
Device not configured (verified if the plugin is installed for this device)


xivo-snom-8.4.31
xivo-aastra-3.2.2.56
xivo-aastra-3.2.2.56
xivo-aastra-3.2.2.56
xivo-aastra-3.2.2.56
Reset to autoprov

Device With one User Only Associated Edit the user associated to the device and put the device field to null.

- click on the `Save` button on the web interface

The phone doesn't restart and the phone is in autoprov mode in the device list.

You can synchronize the device to reboot it.

Device with Several Users Associated Edit the primary user associated to the terminal (one with the line 1) and put the device field to null.

- click on the `Save` button on the web interface

The primary line of the phone has been removed, so the device will lose its funckeys associated to primary user but there others lines associated to the device will stay provisionned.

The phone doesn't restart and the phone is in autoprov mode in the device list.

You can synchronize the device for reboot it.

From a Device

- Dial ***guest** (*48378) on the phone dialpad followed by **xivo** (9486) as a password

The phone restarts and display autoprov, ready to be used for another user.

Advanced Configuration

DHCP Integration

If your phones are getting their network configuration from your XiVO's DHCP server, it's possible to activate the DHCP integration on the *Configuration* → *Provisioning* → *General* page.

What DHCP integration does is that, on every DHCP request made by one of your phones, the DHCP server sends information about the request to `provd`, which can then use this information to update its device database.

This feature is useful for phones which lack information in their TFTP/HTTP requests. For example, without DHCP integration, it's impossible to extract model information for phones from the Cisco 7900 series. Without the model information extracted, there's chance your device won't be automatically associated to the best plugin.

This feature can also be useful if your phones are not always getting the same IP addresses, for one reason or another. Again, this is useful only for some phones, like the Cisco 7900; it has no effect for Aastra 6700.

Creating Custom Templates

Custom templates comes in handy when you have some really specific configuration to make on your telephony devices.

Templates are handled on a per plugin basis. It's not possible for a template to be shared by more than one plugin since it's a design limitation of the plugin system of `provd`.

Note: When you install a new plugin, templates are not migrated automatically, so you must manually copy them from the old plugin directory to the new one. This does not apply for a plugin upgrade.

Let's suppose we have installed the `xivo-aastra-3.2.2-SP3` plugin and want to write some custom templates for it.

First thing to do is to go into the directory where the plugin is installed:

```
cd /var/lib/xivo-provd/plugins/xivo-aastra-3.2.2-SP3
```

Once you are there, you can see there's quite a few files and directories:

```
tree
.
+-- common.py
+-- entry.py
+-- pkgs
|   +-- pkgs.db
+-- plugin-info
+-- README
+-- templates
|   +-- 6730i.tpl
|   +-- 6731i.tpl
|   +-- 6739i.tpl
|   +-- 6753i.tpl
|   +-- 6755i.tpl
|   +-- 6757i.tpl
|   +-- 9143i.tpl
|   +-- 9480i.tpl
|   +-- base.tpl
+-- var
    +-- cache
    +-- installed
    +-- templates
    +-- tftpboot
        +-- Aastra
            +-- aastra.cfg
```

The interesting directories are:

templates This is where the original templates lies. You *should not* edit these files directly but instead copy the one you want to modify in the var/templates directory.

var/templates This is the directory where you put and edit your custom templates.

var/tftpboot This is where the configuration files lies once they have been generated from the templates. You should look at them to confirm that your custom templates are giving you the result you are expecting.

> **Warning:** When you uninstall a plugin, the plugin directory is removed altogether, including all the custom templates.

A few things to know before writing your first custom template:

- templates use the Jinja2 template engine.
- when doing an `include` or an `extend` from a template, the file is first looked up in the `var/templates` directory and then in the `templates` directory.
- device in autoprov mode are affected by templates, because from the point of view of `provd`, there's no difference between a device in autoprov mode or fully configured. This means there's usually no need to modify static files in `var/tftpboot`. And this is a bad idea since a plugin upgrade will override these files.

Custom template for every devices

```
cp templates/base.tpl var/templates
vi var/templates/base.tpl
provd_pycli -c 'devices.using_plugin("xivo-aastra-3.2.2-SP3").reconfigure()'
```

Once this is done, if you want to synchronize all the affected devices, use the following command:

```
provd_pycli -c 'devices.using_plugin("xivo-aastra-3.2.2-SP3").synchronize()'
```

Custom template for a specific model Let's supose we want to customize the template for our 6739i:

```
cp templates/6739i.tpl var/templates
vi var/templates/6739i.tpl
provd_pycli -c 'devices.using_plugin("xivo-aastra-3.2.2-SP3").reconfigure()'
```

Custom template for a specific device To create a custom template for a specific device you have to create a device-specific template named <device_specific_file_with_extension>.tpl in the `var/templates/` directory :

- for an Aastra phone, if you want to customize the file `00085D2EECFB.cfg` you will have to create a template file named `00085D2EECFB.cfg.tpl`,
- for a Snom phone, if you want to customize the file `000413470411.xml` you will have to create a template file named `000413470411.xml.tpl`,
- for a Polycom phone, if you want to customize the file `0004f2211c8b-user.cfg` you will have to create a template file named `0004f2211c8b-user.cfg.tpl`,
- and so on.

Here, we want to customize the content of a device-specific file named `00085D2EECFB.cfg`, we need to create a template named `00085D2EECFB.cfg.tpl`:

```
cp templates/6739i.tpl var/templates/00085D2EECFB.cfg.tpl
vi var/templates/00085D2EECFB.cfg.tpl
provd_pycli -c 'devices.using_mac("00085D2EECFB").reconfigure()'
```

Note: The choice to use this syntax comes from the fact that `provd` supports devices that do not have MAC addresses, namely softphones.

Also, some devices have more than one file (like Snom), so this way make it possible to customize more than 1 file.

The template to use as the base for a device specific template will vary depending on the need. Typically, the model template will be a good choice, but it might not always be the case.

Changing the Plugin Used by a Device

From time to time, new firmwares are released by the devices manufacturer. This sometimes translate to a new plugin being available for these devices.

When this happens, it almost always means the new plugin obsoletes the older one. The older plugin is then considered "end-of-life", and won't receive any new updates nor be available for new installation.

Let's suppose we have the old `xivo-aastra-3.2.2.1136` plugin installed on our xivo and want to use the newer `xivo-aastra-3.2.2-SP3` plugin.

Both these plugins can be installed at the same time, and you can manually change the plugin used by a phone by editing it via the *Services → IPBX → Devices* page.

If you are using custom templates in your old plugin, you should copy them to the new plugin and make sure that they are still compatible.

Once you take the decision to migrate all your phones to the new plugin, you can use the following command:

```
provd_pycli -c 'helpers.mass_update_devices_plugin("xivo-aastra-3.2.2.1136", "xivo-aastra-3.2.2-S
```

Or, if you also want to synchronize (i.e. reboot) them at the same time:

```
provd_pycli -c 'helpers.mass_update_devices_plugin("xivo-aastra-3.2.2.1136", "xivo-aastra-3.2.2-S
```

You can check that all went well by looking at the *Services → IPBX → Devices* page.

Remote directory

If you have a phone provisioned with XiVO and its one of the supported ones, you'll be able to search in your XiVO directory and place call directly from your phone.

Supported devices

Tested

- Aastra 6700 series using the **3.2** or later firmware (does not work on firmware 2.6)
- Snom 320
- Cisco 7940G, 7941G

Untested

- Thomson
- Yealink

Configuration

For the remote directory to work on your phones, the first thing to do is to go to the *Services → IPBX → (General settings) Phonebook* page.

You then have to add the range of IP addresses that will be allowed to access the directory. So if you know that your phone's IP addresses are all in the 192.168.1.0/24 subnet, just click on the small "+" icon and enter "192.168.1.0/24", then save.

Once this is done, on your phone, just click on the "remote directory" function key and you'll be able to do a search in the XiVO directory from it.

Jitsi

Jitsi (http://jitsi.org/) is an opensource softphone (previously SIP Communicator).

XiVO now support Jitsi sofphones provisioning. Here are the steps to follow :

Requirements

This how to needs :

1. Jitsi installed,
2. SIP line created

Add Jitsi plugin on XiVO

Open XiVO Web interface, and go to Configuration tab, Then chose *Provisioning → Plugins menu*, Install the Jitsi plugin you want to use : e.g.:

```
xivo-jitsi-1
```

You can now launch your Jitsi softphone

Configuring Jitsi

1. Launch Jitsi,
2. If you don't have any accounts configured Jitsi will launch a windows and you can click
3. Use online provisioning. Otherwise go to Tools -> Options -> Advanced -> Provisioing, Click on Enable provisioning
4. Select Manually specify a provisioning URI,
5. Enter the folowing URI where <provd_ip> is the VoIP interface IP address of your XiVO and <provd_port> is the provd port (default : 8667)

   ```
   http://<provd_ip>:<provd_port>/jitsi?uuid=${uuid}
   ```

6. When done, quit Jitsi,
7. Launch Jitsi again,

- You should now be connected with in autoprov mode,
- You could see a new device in the devices list,

8. You can now provision the phones by typing the provisioning code (you get it in the Lines list),
9. Quit Jitsi again (configuration syncing is not available with the Jitsi plugin)
10. And launch Jitsi again : you should now be connected with you phone account

1.8.24 SCCP Configuration

Activating DHCP Server: *Configuration → Network → DHCP*

Activating DHCP Integration: *Configuration → Provisioning → General*

Installing a plugin for SCCP Phone: *Configuration → Provisioning → Plugins*

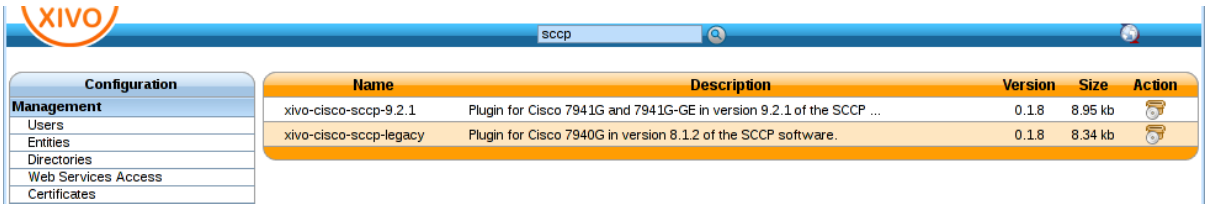


Figure 1.56: Installing xivo cisco-sccp plugin

Review SCCP general settings: *Services → IPBX → IPBX settings → SCCP general settings*

At this point you should have a fully functional DHCP server that provides IP address to your phones. Depending on what type of CISCO phone you have, you need to install the plugin sccp-legacy, sccp-9.0.3 or both. Please refer to the *Provisioning page* for more information on how to install CISCO firmwares.

Once your plugin is installed, you'll be able to edit which firmwares and locales you need. If you are unsure, you can choose all without any problem.

Now if you connect your first SCCP phone, you should be able to see it in the device list.

Listing the detected devices: *Services → IPBX → IPBX settings → Devices*

When connecting a second SCCP phone, the device will be automatically detected as well.

The last step is to create a user with a SCCP line.

Creating a user with a SCCP line: *Services → IPBX → IPBX settings → Users*

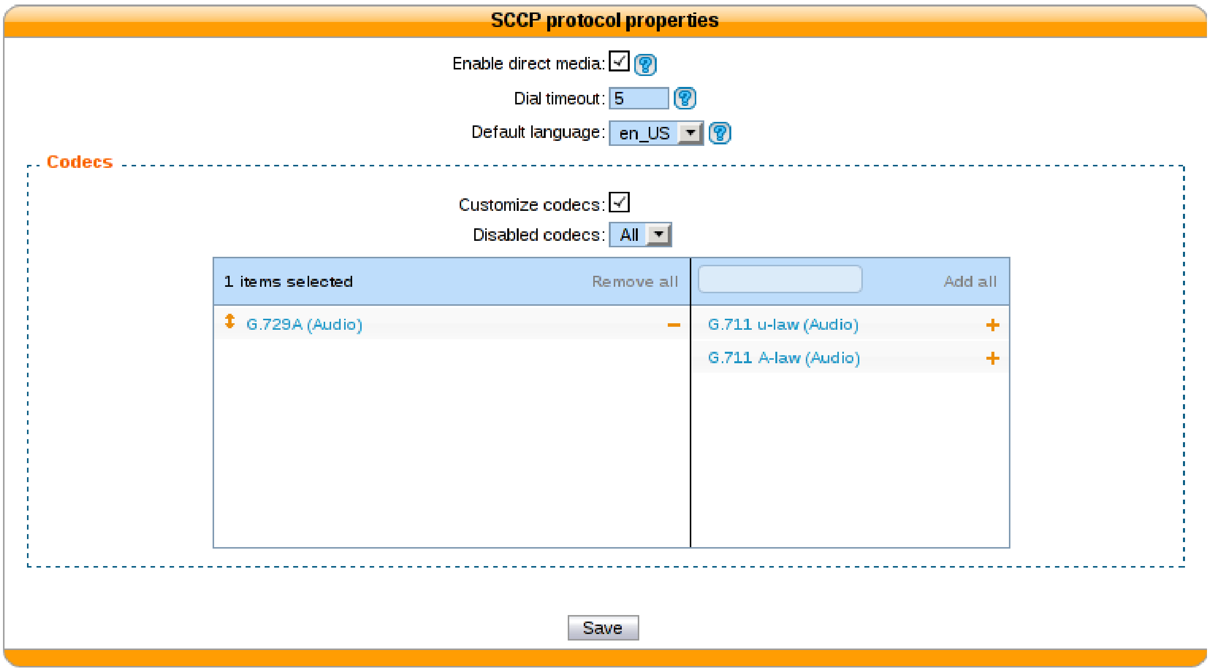


Figure 1.57: SCCP general settings

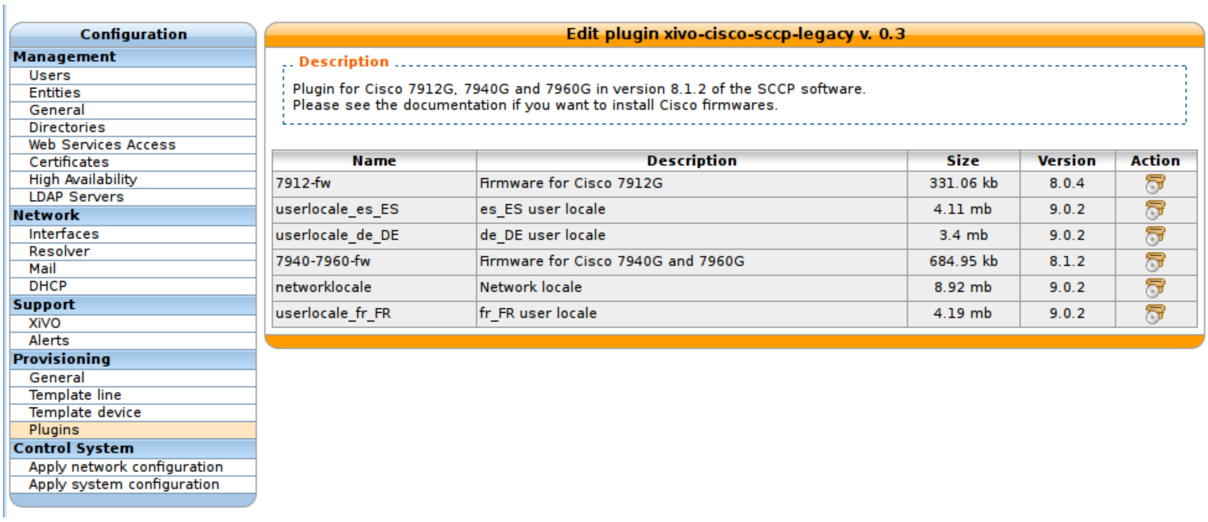


Figure 1.58: Editing the xivo-cisco-sccp-legacy plugin

Figure 1.59: Device list

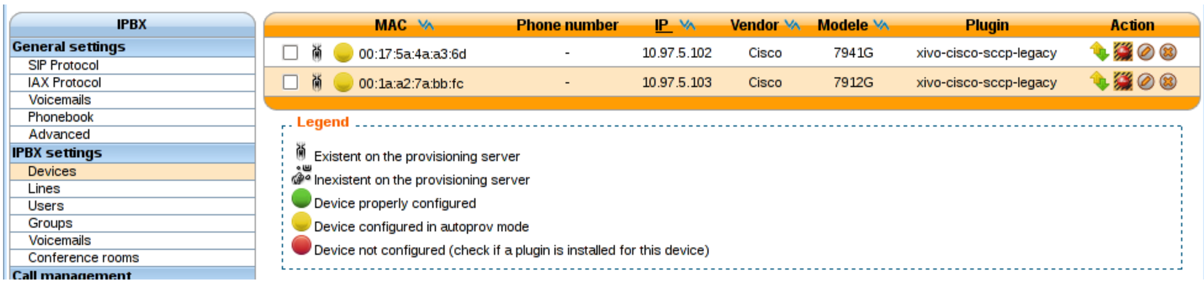


Figure 1.60: Device list

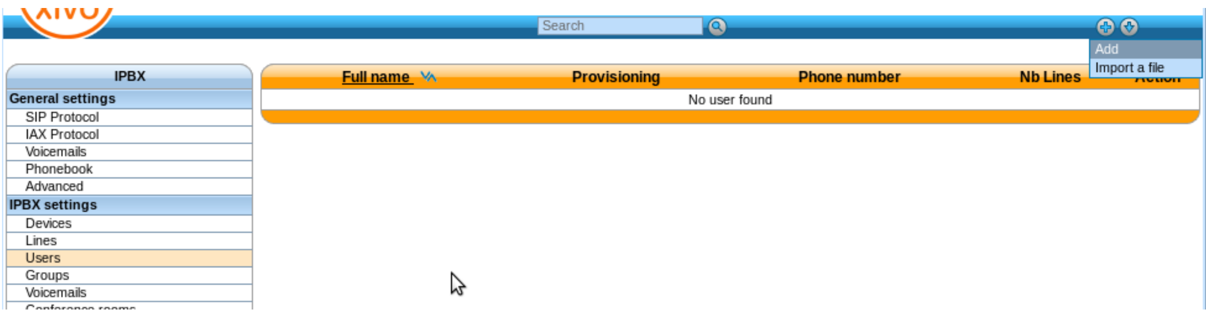


Figure 1.61: Add a new user

Figure 1.62: Edit user informations

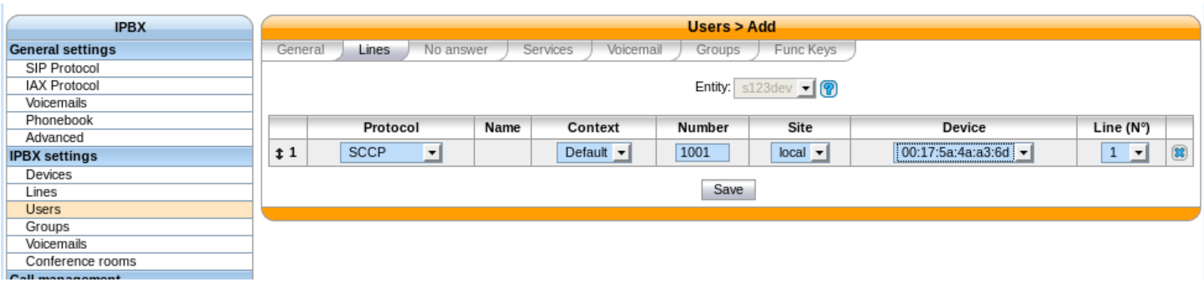


Figure 1.63: Add a line to a user

Before saving the newly configured user, you need to select the *Lines* menu and add a SCCP line. Now, you can save your new user.

Congratulations ! Your SCCP phone is now ready to be called !

Direct Media

SCCP Phones support directmedia (direct RTP). In order for SCCP phones to use directmedia, one must enable the directm

Services → IPBX → IPBX settings → SCCP general settings

Features

Features	Supported
Receive call	Yes
Initiate call	Yes
Hangup call	Yes
Transfer call	Yes
Congestion Signal	Yes
Autoanswer (custom dialplan)	Yes
Call forward	Yes
Multi-instance per line	Yes
Message waiting indication	Yes
Music on hold	Yes
Context per line	Yes
Paging	Yes
Direct RTP	Yes
Redial	Yes
Speed dial	Yes
BLF (Supervision)	Yes
Resync device configuration	Yes
Do not disturb (DND)	Yes
Group listen	Yes
Caller ID	Yes
Connected line ID	Yes
Group pickup	Not yet
Auto-provisioning	Not yet
Multi line	Not yet
Codec selection	Yes
NAT traversal	Not yet
Type of Service (TOS)	Manual

Telephone

Device type	Supported	Firmware version	Direct media
7905	Should work		Yes
7906	Should work		Yes
7911	Yes	SCCP11.8-5-3S	Yes
7912	Yes	8.0.4(080108A)	Yes
7920	Yes	3.0.2	Yes
7921	Yes	1.4.5.3	Yes
7940	Yes	8.1(2.0)	Yes
7941	Yes	SCCP41.9-0-3S	Yes
7941GE	Yes	SCCP41.9-0-3S	Yes
7942	Yes	SCCP42.9-0-3S	Yes
7960	Yes	8.1(2.0)	Yes
7961	Yes	SCCP41.9-0-3S	Yes
7962	Yes	SCCP42.9-0-3S	Yes
CIPC	Yes	2.1.2	Yes

An unsupported device won't be able to connect to Asterisk (channel sccp) at all.

Hand written configuration

The sccp.conf.sample file can be consulted for an example of an hand written configuration file.

1.8.25 Schedules

Schedules are specific time frames that can be defined to open or close a service. Within schedules you may specify opening days and hours or close days and hours.

A default destination as user, group ... can be defined when the schedule is in closed state.

Schedules can be applied to :

- Users
- Groups
- Inbound calls
- Outbound calls
- Queues

Creating Schedules

A schedule is composed of a name, a timezone, one or more opening hours or days that you may setup using a calendar widget, a destination to be used when the schedule state is closed.

With the calendar widget you may select months, days of month, days of week and opening time.

You may also optionaly select closed hours and destination to be applied when period is inside the main schedule. For example, your main schedule is opened between 08h00 and 18h00, but you are closed between 12h00 and 14h00.

Using Schedule on Users

When you have a schedule associated to a user, if this user is called during a closed period, the caller will first hear a prompt saying the call is being transferred before being actually redirected to the closed action of the schedule.

If you don't want this prompt to be played, you can change the behaviour by:

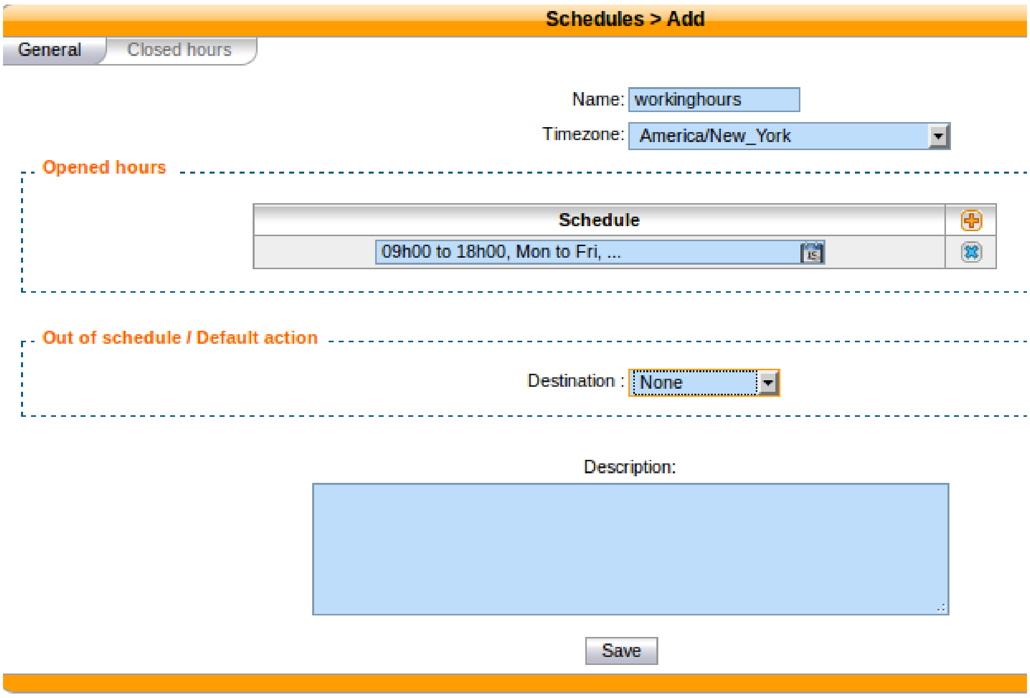


Figure 1.64: Creating a schedule

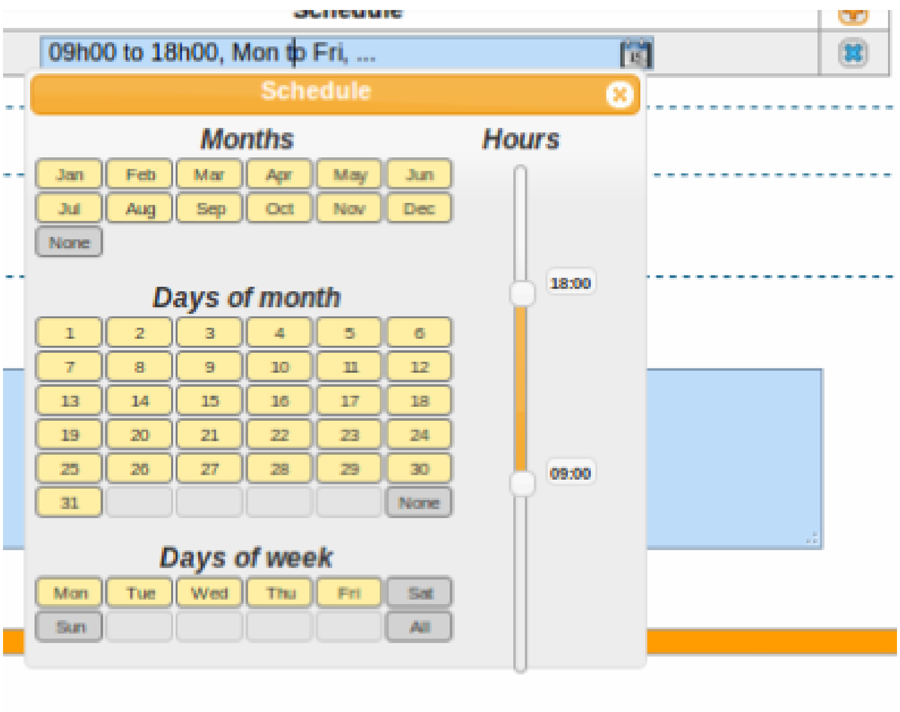


Figure 1.65: Schedule calendar widget

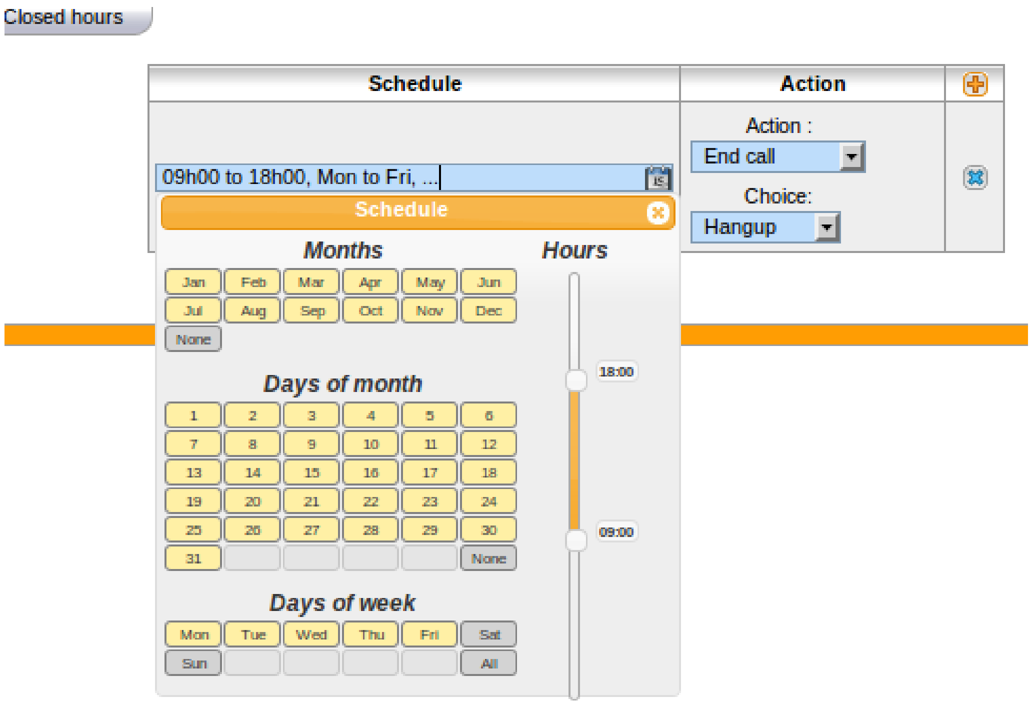


Figure 1.66: Schedule closed hours

1. editing the `/etc/xivo/asterisk/xivo_globals.conf` file and setting the `XIVO_FWD_SCHEDULE_OUT_ISDA` to `1`
2. reloading the asterisk dialplan with an `asterisk -rx "dialplan reload"`.

1.8.26 Sound Files

Add Sounds Files

On a fresh install, only en_US and fr_Fr sounds are installed. Canadian French and German are available too.

To install Canadian French sounds you have to execute the following command in the cli:

```
root@xivo:~# apt-get install asterisk-sounds-wav-fr-ca xivo-sounds-fr-ca
```

To install German sounds you have to execute the following command in the cli:

```
root@xivo:~# apt-get install asterisk-sounds-wav-de-de xivo-sounds-de-de
```

Now you may select the newly installed language for yours users.

Convert Your Wav File

Asterisk will read natively WAV files encoded in wav 8kHz, 16 bits, mono.

The following command will return the encoding format of the <file>

```
$ file <file>
RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz
```

The following command will re-encode the <input file> with the correct parameters for asterisk and write into the <output file>

```
$ sox <input file> -b 16 -c 1 -r 8000 -t wavpcm <output file>
```

1.8.27 Switchboard

This page describes the configuration needed to have a switchboard on your XiVO.

Overview

Switchboard functionality is available in the XiVO client. The goal of this page is to explain how to configure your switchboard and how to use it.

The switchboard xlet and profile allow an operator to view incoming calls, answer them, put calls on hold, view the calls on hold and pick up the calls on hold.

Limitations

Note: The shortcut keys of the switchboard do not work on the Mac version of the XiVO client.

Note: The enter shortcut to answer a call will not work if the focus is currently on a widget that will consume the key press. ie: a text field, a drop down list or a button.

Configuration

Quick Summary

In order to configure a switchboard on your XiVO, you need to:

- Create a queue for your switchboard
- Create a queue for your switchboard's calls on hold
- Create the users that will be operators
- Activate the switchboard option for your phone
- Create an agent for your user
- Assign the incoming calls to the switchboard queue
- For each operator, add a function key for logging in or logging out from the switchboard queue.
- Set "no answer" destinations on the switchboard queue

Supported Devices

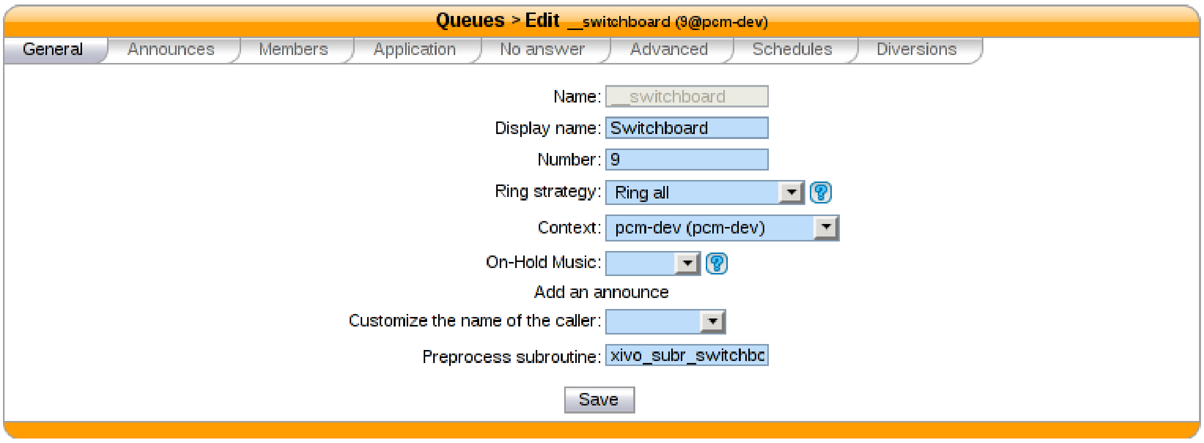
The supported phones for the switchboard are:

- Aastra 6755i
- Aastra 6757i
- Snom 720

Create a Queue for Your Switchboard

All calls to the switchboard will first be distributed to a switchboard queue.

To create this queue, go to *Services* → *Call center* → *Queues* and click the add button.



The Following configuration is mandatory

- The *General* → *Name* field has to be *__switchboard*
- The *General* → *Preprocess subroutine* field has to be *xivo_subr_switchboard*
- The *Application* → *Allow caller to hang up call* option has to be *enabled*
- The *Application* → *Allow callee to transfer the call* option has to be *enabled*
- The *Advanced* → *Member reachability timeout* option has to be *disabled*
- The *Advanced* → *Time before retrying a call to a member* option has to be *1 second*
- The *Advanced* → *Delay before reassigning a call* option has to be *disabled*
- The *Advanced* → *Call a member already on* option has to be *disabled*
- The *Advanced* → *Autopause agents* option has to be *disabled*

Other important fields

- The *General* → *Display name* field is the name displayed in the XiVO client xlets and in the statistics
- The *General* → *Number* field is the number that will be used to reach the switchboard internally (typically *9*)

Create a Queue for Your Switchboard on Hold

The switchboard uses a queue to track its calls on hold.

To create this queue, go to *Services* → *Call center* → *Queues* and click the add button.

The Following configuration is mandatory

- The *General* → *Name* field has to be *__switchboard_hold*
- The *General* → *Number* field has to be a valid number in a context reachable by the switchboard

Other important fields

- The *General* → *Display name* field is the name displayed in the XiVO client xlets and in the statistics

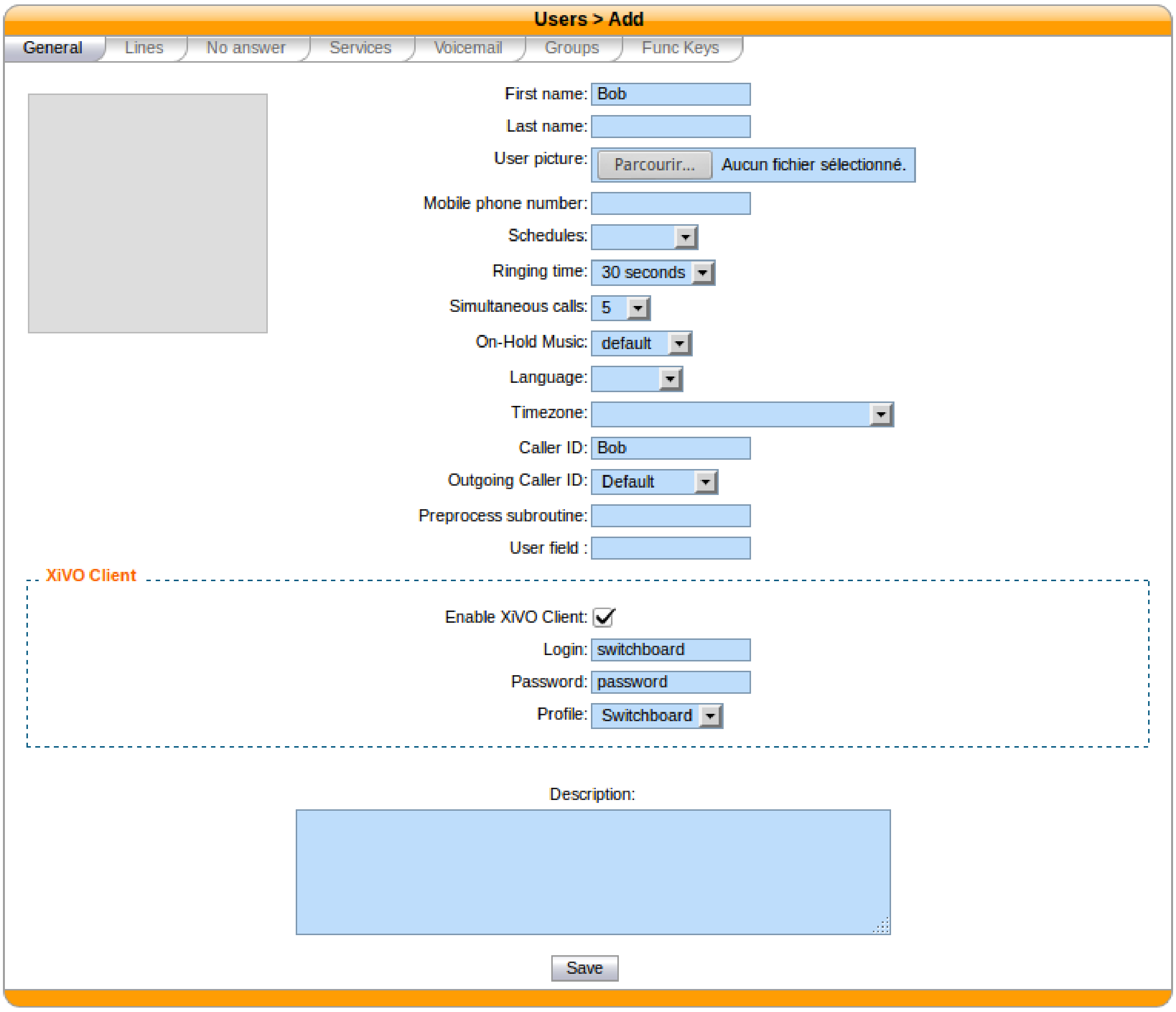
> **Warning:** This queue MUST have **NO** members

Create the Users that Will be Operators

Each operator needs to have a user configured with a line. The XiVO client profile has to be set to *Switchboard*.

The following configuration is mandatory for switchboard users

- The *General* → *First name* field has to be set
- The *General* → *Enable XiVO Client* option has to be *enabled*
- The *General* → *Login* field has to be set
- The *General* → *Password* field has to be set
- The *General* → *Profile* field has to be set to *Switchboard*
- The *Lines* → *Number* field has to have a valid extension
- The *Lines* → *Device* field has to be a *supported device*
- The *Services* → *Enable call transfer* option has to be *enabled*

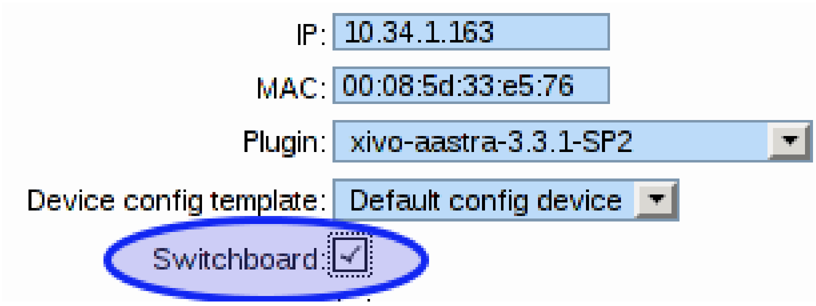


Activate the Switchboard Option for your Phone

For the switchboard to work properly, your Aastra or Snom phone must use a *xivo-aastra* or *xivo-snom* provisioning plugin respectively.

The switchboard option must also be activated on the phone. It's possible to activate this option only on supported phones and plugins.

- Edit device associated to your user in *Services* → *Devices*
- Check the switchboard checkbox and save
- Synchronize your phone to apply the changes



Warning: To be able to use a Snom phone for the switchboard, you have to be able to do the appropriate HTTP request from the XiVO to the device's web service. The following command should work from your XiVO's bash command line *wget http://guest:guest@<phone IP address>/command.htm?key=SPEAKER*. If this command does not activate the phone's speaker, your network configuration will have to be *fixed* before you can be able to use the Snom switchboard.

Warning: When using a Snom switchboard you should not use the first function key.

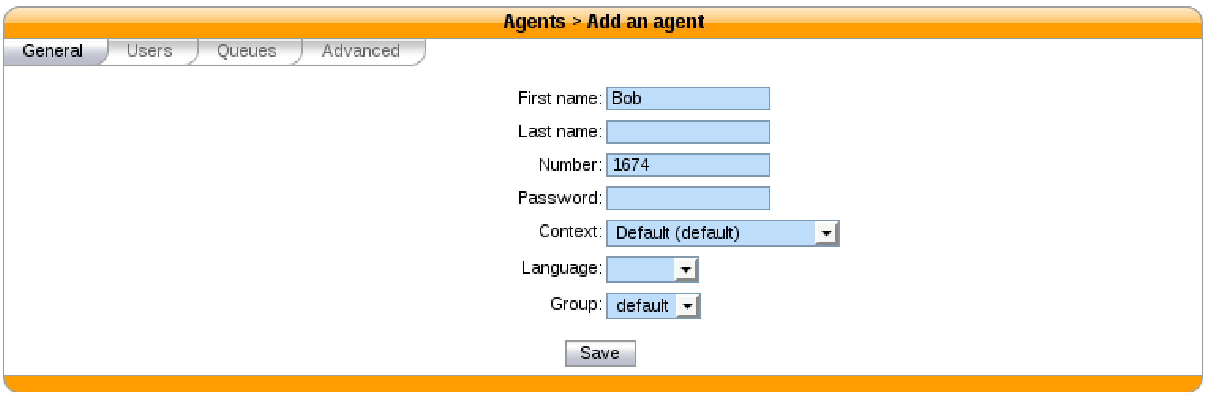
Create an Agent for the Operator

Each operator needs to have an associated agent.

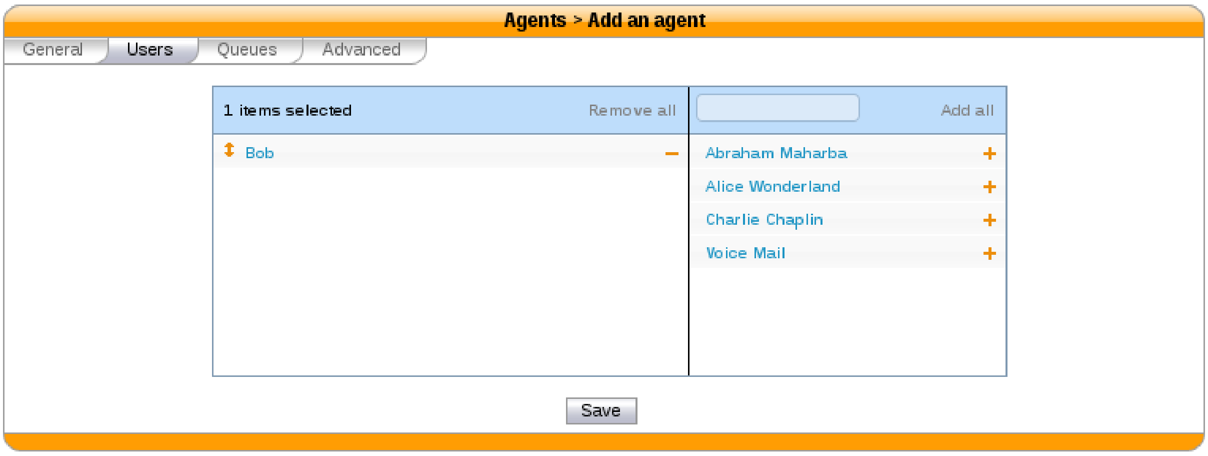
Warning: Each agent MUST ONLY be a member of the Switchboard queue

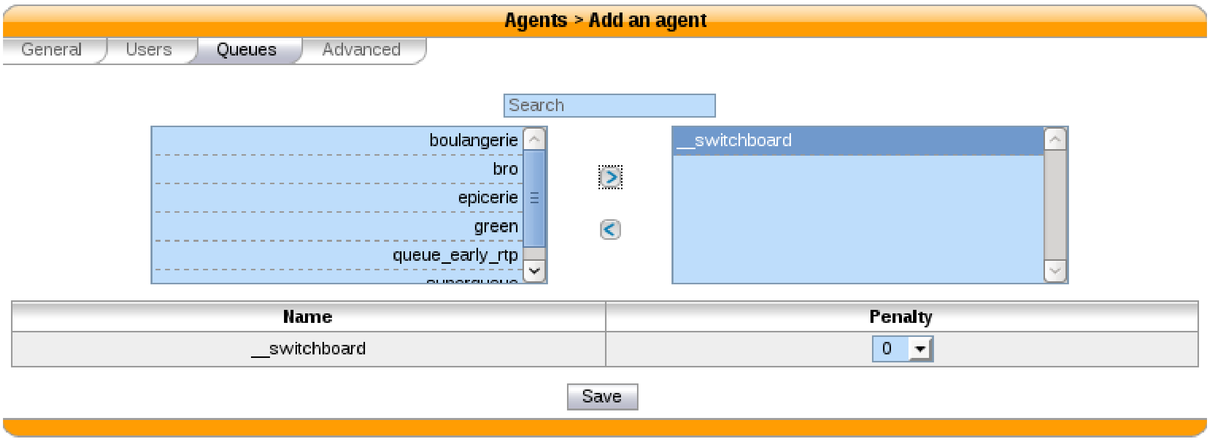
To create an agent:

- Go to *Services* → *Call center* → *Agents*
- Click on the group *default*
- Click on the *Add* button



- Associate the user to the agent in the *Users* tab
- Assign the Agent to the *Switchboard* Queue (**and ONLY to the Switchboard queue**)





Send Incoming Calls to the *Switchboard* Queue

Incoming calls must be sent to the *Switchboard* queue to be distributed to the operators. To do this, we have to change the destination of our incoming call for the switchboard queue.

In this example, we associate our incoming call (DID *444*) to our *Switchboard* queue:

Set "No Answer" Destinations on the *Switchboard* Queue

When there are no operators available to answer a call, "No Answer" destinations should be used to redirect calls towards another destination.

You also need to set the timeout of the Switchboard queue to know when calls will be redirected.

The reachability timeout must not be disabled nor be too short.

The time before retrying a call to a member should be as low as possible (1 second).

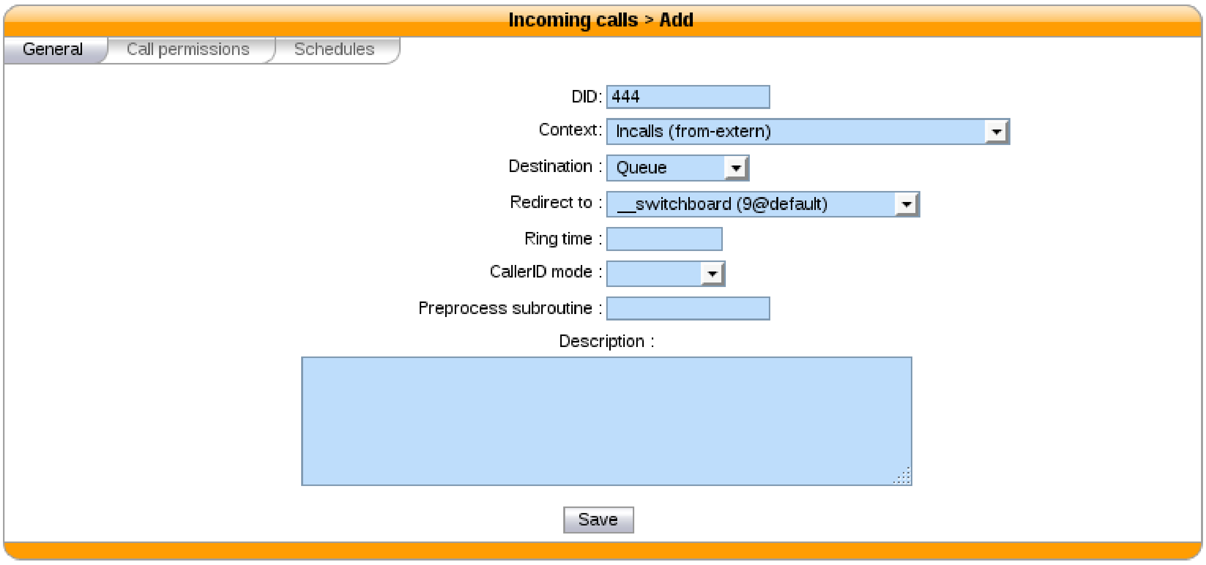
In this example we redirect "No Answer", "Busy" and "Congestion" calls to the *everyone* group and "Fail" calls to the *guardian* user.

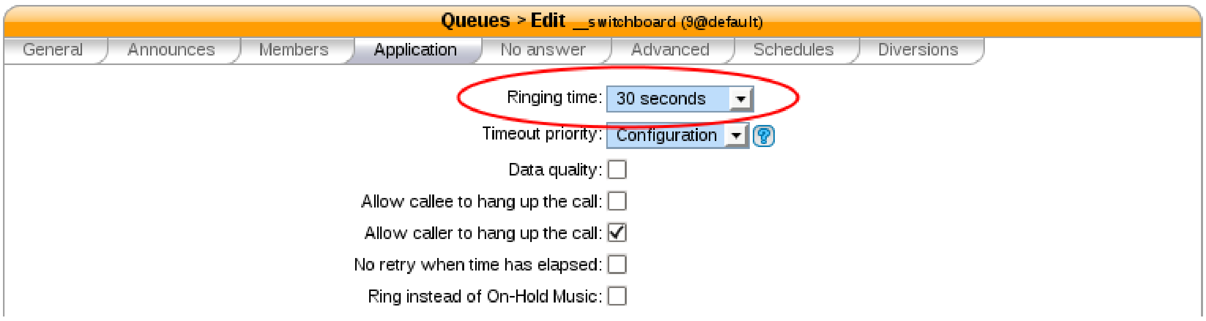
You can also choose to redirect all the calls to another user or a voice mail.

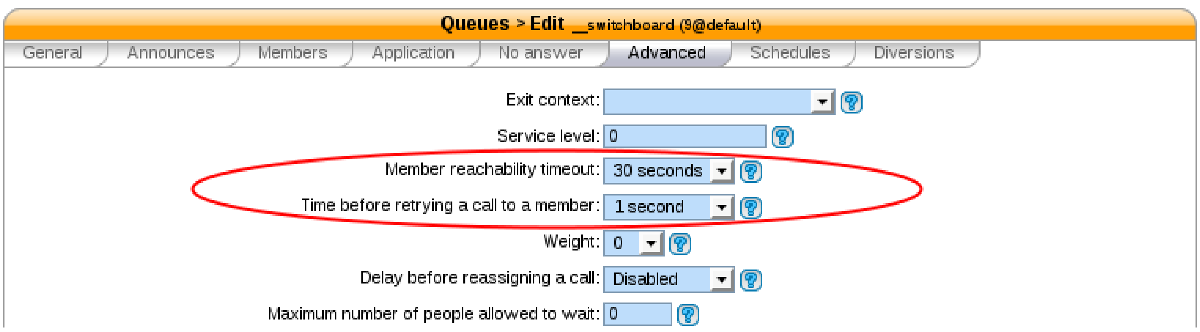
XiVO Client configuration

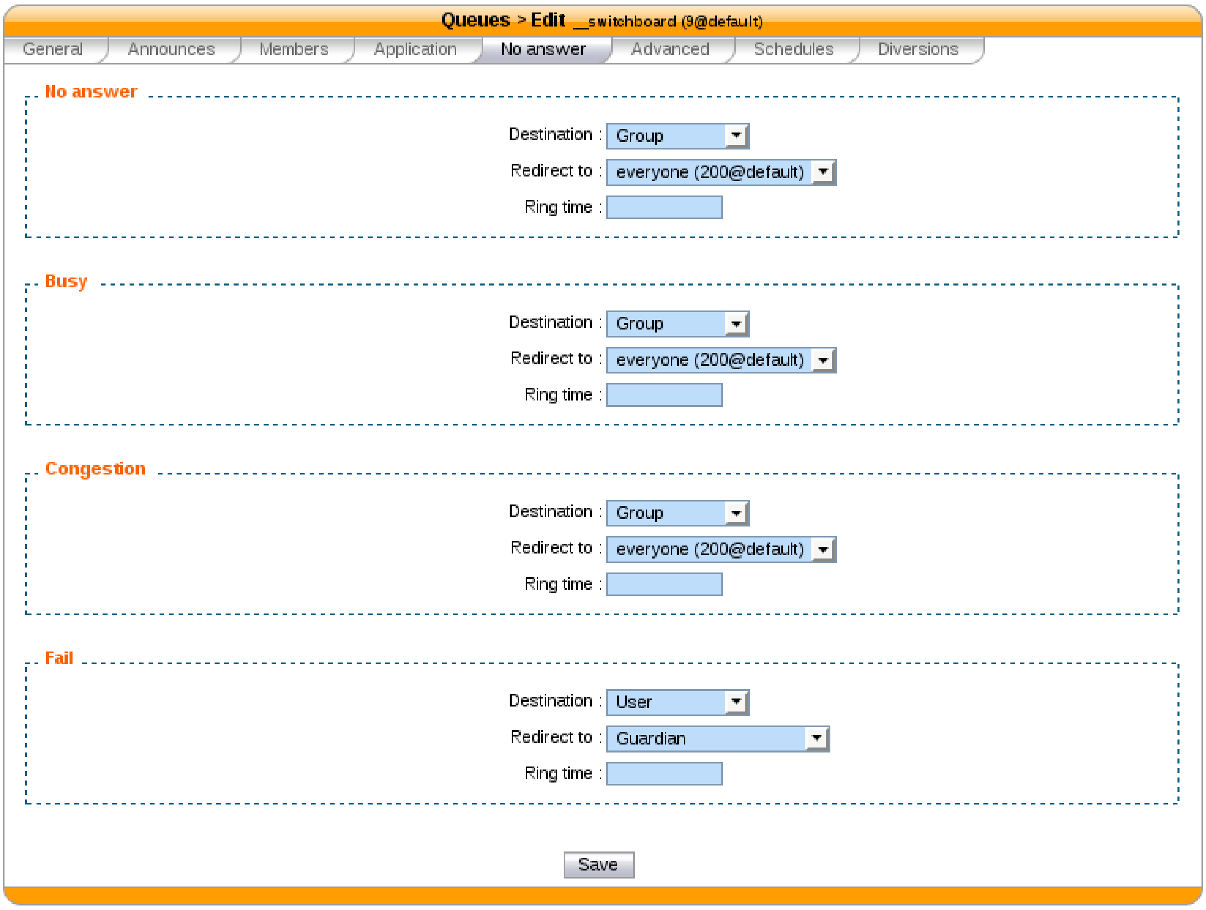
Directory xlet

The transfer destination is chosen in the Directory xlet. You **must** follow the *Directory* section to be able to use it.


Incoming calls > Add
General
Call permissions
Schedules
DID: 444
Context: Incalls (from-extern)
Destination : Queue
Redirect to : __switchboard (9@default)
Ring time :
CallerID mode :
Preprocess subroutine :
Description :
Save


Queues > Edit __switchboard (9@default)
General
Announces
Members
Application
No answer
Advanced
Schedules
Diversions
Ringing time: 30 seconds
Timeout priority: Configuration
Data quality:
Allow callee to hang up the call:
Allow caller to hang up the call:
No retry when time has elapsed:
Ring instead of On-Hold Music:


Queues > Edit __switchboard (9@default)
General
Announces
Members
Application
No answer
Advanced
Schedules
Diversions
Exit context:
Service level: 0
Member reachability timeout: 30 seconds
Time before retrying a call to a member: 1 second
Weight: 0
Delay before reassigning a call: Disabled
Maximum number of people allowed to wait: 0



Configuration for multiple switchboards

The above documentation can be used for multiple switchboards on the same XiVO by replacing the *__switchboard* and *__switchboard_hold* queues name and configuring the operators XiVO client accordingly in the *XiVO Client → Configure → Functions → Switchboard* window.

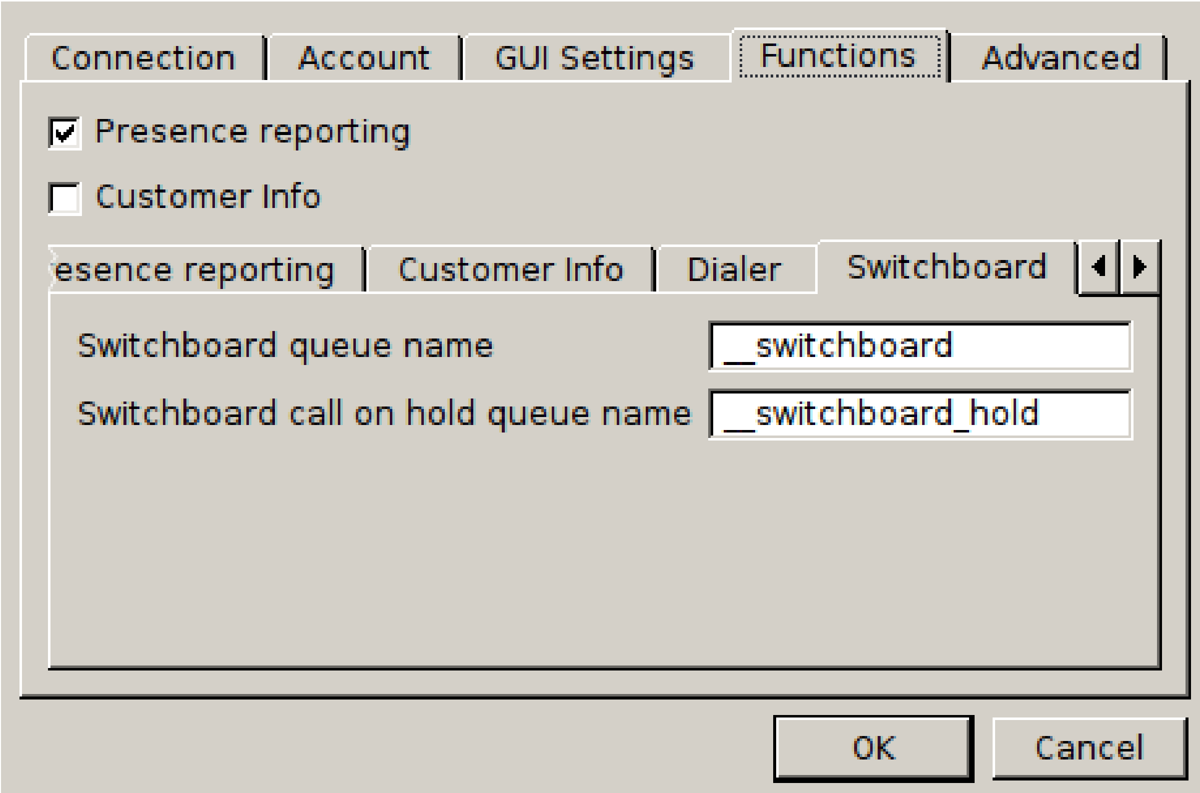
Usage

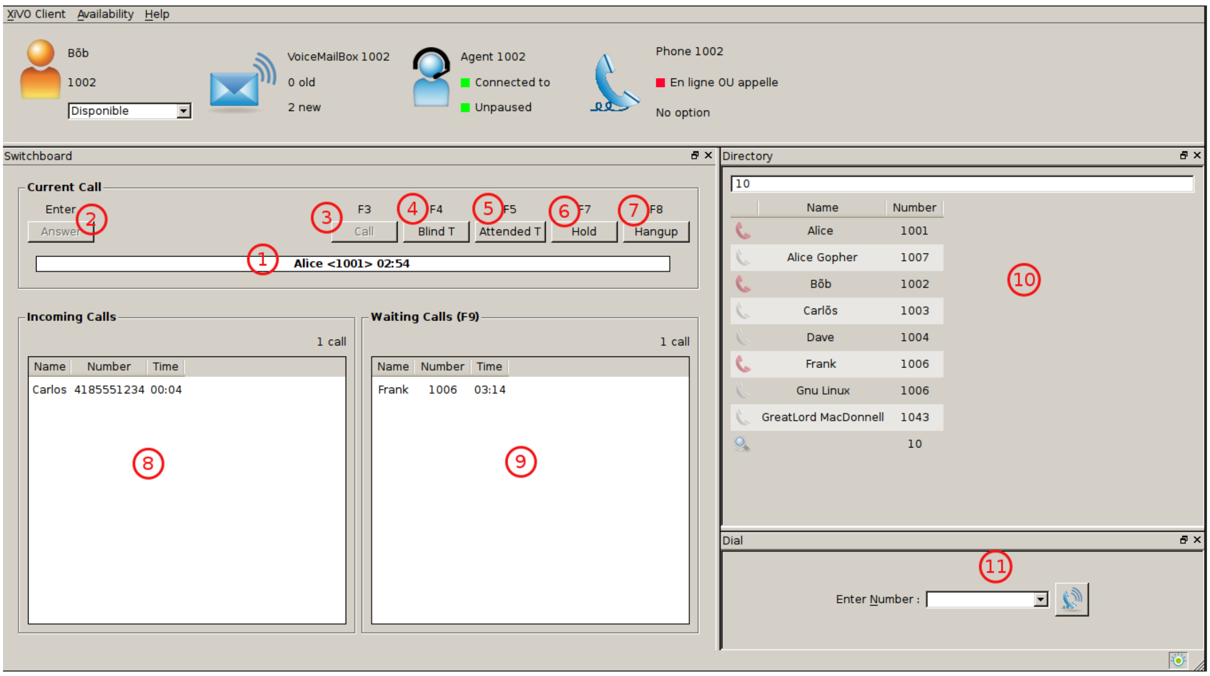
> **Warning:** The switchboard configuration must be completed before using the switchboard. This includes :
> - Device, User, Agent and Queues configuration (see above),
> - Directory xlet configuration (see *Directory*)
>
> If it's not the case, the user must disconnect his XiVO client and reconnect.

The XiVO Client Switchboard Profile

When the user connects with his XiVO Client, he gets the Switchboard profile.

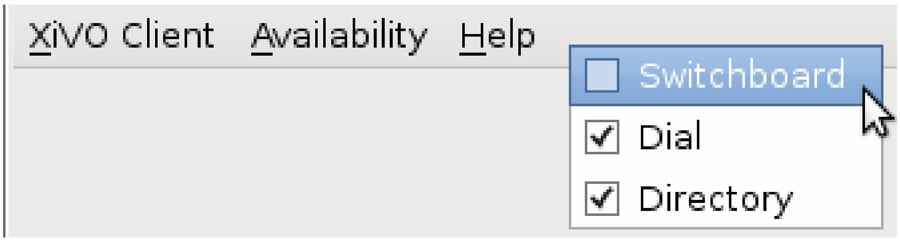
1. *Current Call* frame
2. *Answer* button
3. *Call* button
4. *Blind transfer* button
5. *Attended transfer* button
6. *Hold* button


Connection
Account
GUI Settings
Functions
Advanced
Presence reporting
Customer Info
esence reporting
Customer Info
Dialer
Switchboard
Switchboard queue name
__switchboard
Switchboard call on hold queue name
__switchboard_hold
OK
Cancel


XiVO Client Availability Help
Bôb
1002
Disponible
VoiceMailBox 1002
0 old
2 new
Agent 1002
Connected to
Unpaused
Phone 1002
En ligne OU appelle
No option
Switchboard
Current Call
Enter
Answer
F3 Call
F4 Blind T
F5 Attended T
F7 Hold
F8 Hangup
Alice <1001> 02:54
Incoming Calls
1 call
Name Number Time
Carlos 4185551234 00:04
Waiting Calls (F9)
1 call
Name Number Time
Frank 1006 03:14
Directory
10
Name Number
Alice 1001
Alice Gopher 1007
Bôb 1002
Carlôs 1003
Dave 1004
Frank 1006
Gnu Linux 1006
GreatLord MacDonnell 1043
10
Dial
Enter Number :
1
2
3
4
5
6
7
8
9
10
11

7. *Hangup* button
8. *Incoming Calls* list
9. *Waiting Calls* list
10. *Directory* Xlet
11. *Dial* Xlet

Note: If you don't see the Switchboard Xlet, right-click on the grey bar at the right of the *Help* menu and check *Switchboard*:



The operator can login his agent using a function key or an extension to start receiving calls.

Call flow

Answering an incoming call When the switchboard receives a call, the new call is added to the *Incoming Calls* list on the left and the phone starts ringing. The user can answer this call by:

- clicking on any call in the list
- clicking the *Answer* button
- pressing the *Enter* key

Note: The XiVO Client must be the active window for the keyboard shortcuts to be handled

The operator can select which call to answer by:

- clicking directly on the incoming call
- pressing *F6* to select the incoming calls frame and pressing the up and down arrow keys

Selecting a call to answer while talking will not answer the call.

Once the call has been answered, it is removed from the incoming calls list and displayed in the *Current Call* frame.

Making a Call The switchboard operator can do the following operations:

- Press the *Call* button or press *F3*
- Search for the call destination in the directory xlet
- Press to confirm the selection and start the call

Hanging Up a Call The switchboard operator can hang up its current call by either:

- Clicking the *Hangup* button
- Pressing the *F8* key

If the operator has placed a new call via the *Directory* or *Dial* xlet and that call has not yet been answered, he can cancel it in the same way.

Distributing a call Once the call has been answered and placed in the current call frame, the operator has 3 choices:

- transfer the call to another user
  - using the *Blind transfer* button or the *F4* key.
  - using the *Attended transfer* button or the *F5* key
- put the call on hold using the *Hold* button or the *F7* key
- end the call using the *Hangup* button or the *F8* key.

Transferring a call Transfer buttons allow the operator to select towards which destination he wishes to transfer the call. This is made through the *Directory* xlet. For defails about the xlet *Directory* usage and configuration see *Directory*.

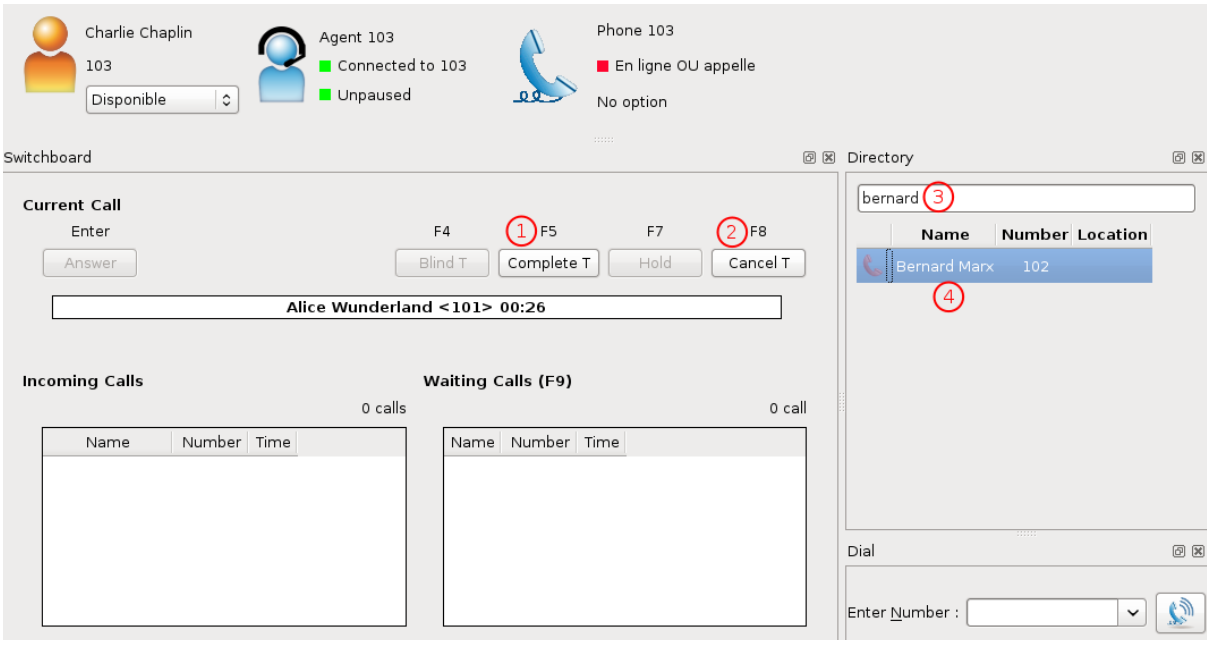
Once the destination name has been entered, press *Enter*. If multiple destinations are displayed, you can choose by:

- double-clicking on the destination
- using *Up/Down* arrows then:
  - pressing *Enter*
  - pressing the transfer button again

Blind transfers are straightforward: once the call is transferred, the operator is free to manage other calls.

Attended transfers are a bit more complicated: the operator needs to wait for the transfer destination to answer before completing the transfer.

In this example, the operator is currently asking *Bernard Marx* if he can transfer *Alice Wonderland* to him.



1. *Complete transfer* button
2. *Cancel transfer* button
3. Transfer destination filtering field (xlet *Directory*)
4. Transfer destination list (xlet *Directory*)

Once the destination has answered, you can:

- cancel the transfer with *F8* key
- complete the transfer with *F5* key

Note: The operator can not complete an attended transfer while the transfer destination is ringing. In this case, the operator must cancel the attended transfer and use the *Blind transfer* action.

Putting a call on hold If the user places the call on hold, it will be removed from the *Current call* frame and displayed in the *Waiting calls* list. The time counter shows how long the call has been waiting, thus it will be reset each time the call returns in the *Waiting calls* list. The calls are ordered from the oldest to the newest.

Retrieving a call on hold Once a call has been placed on hold, the operator will most certainly want to retrieve that call later to distribute it to another destination.

To retrieve a call on hold:

- click the desired call in the *Waiting calls* list
- with the keyboard:
  - move the focus to the *Waiting calls* list (*F9* key)
  - choose the desired call with the arrow keys
  - press the *Enter* key.

Once a call has been retrieved from the *Waiting calls* list, it is moved back into the *Current Call* frame, ready to be distributed.

1.8.28 Users

Users Configuration.

Importing Users

You may import your users using a csv comma separated file. Users and lines are automatically created.

How to import users

Once you have saved your file, you can import your users via the *Services* → *IPBX* → *IPBX settings* → *Users* page by clicking on the plus button.

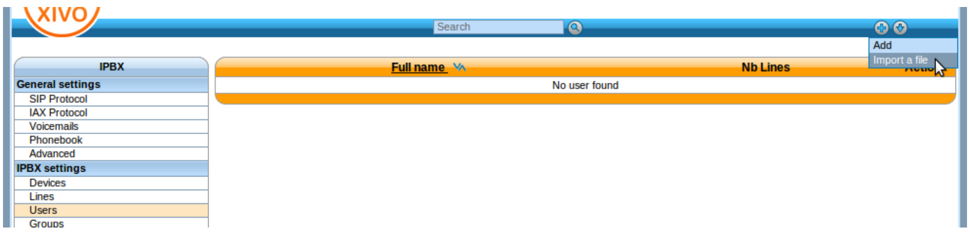


Figure 1.67: Import Users

Supported fields

Field	Values	Description
[section user]	**To add a user**	
entityid	int	entity id (configuration menu) `Must be a val`
firstname *	string	User firstname
lastname	string	User lastname
language **	enum *['de_DE', 'en_US', 'es_ES', 'fr_FR', 'fr_CA']*	Locale `Must be set if you add a voi`
enableclient	bool *[0, 1]*	If set to 1, username and password fields have to b
username	string	XiVO Client username
password	string	XiVO Client password
profileclient	string	XiVO Client profile defined in menu: *Services* > (
outcallerid	string	Customize outgoing caller id for this user
agentnumber	string	Associated agent number
mobilephonenumber	string	Mobile phone number
bosssecretary	enum *['no','boss','secretary']*	Filter: Boss - Secretary
enablehint	bool *[0, 1]*	Enable/Disable supervision
enablexfer	bool *[0, 1]*	Enable/Disable call transfers
[section line]	**To add a line to an user**	
phonenumber *	string	User phone number creates a line `Must exist`
context *	string	context name `internal context must ex`
protocol *	enum *['sip', 'sccp']*	Line protocol
linename	string	Line name (SIP only)
linesecret	string	Line secret (SIP only)
[section incall]	**To add an incall to an user**	
incallexten *	string	DID number `incallexten must exist i`
incallcontext *	string	Context name `incall context must exis`
incallringseconds	int	Ring time in seconds
[section voicemail]	**To add a voicemail to a user**	You must set a language to use this section
voicemailname *	string	Voicemail fullname
voicemailmailbox *	string	Mailbox number
voicemailpassword	string	Password voicemail
voicemailemail	string	Mail to send a notification when a message is rece
voicemailattach	bool *[0, 1]*	Enable/Disable attach the audio file to your mail
voicemaildelete	bool *[0, 1]*	Enable/Disable delete message after notification
voicemailskippass	bool *[0, 1]*	Enable/Disable password checking

Warning: "*", this field is required - valid by section

Warning: "**", this field is required if you add a voicemail

Examples

First step is to create a text file containing the users you want to create. Here's a basic example:

```
entityid|firstname|lastname|phonenumber|context|protocol|mobilephonenumber
1|John|Doe|1000|default|sip|00123456789
1|George|Clinton|1001|default|sip|00123456789
1|Bill|Bush|1002|default|sip|00123456789
```

This example defines 3 users:

- John Doe with one SIP line with number 1000

- George Clinton with one SIP line with number 1001
- Bill Bush with one SIP line with number 1002

Note: Note that the number you use must all be in the range you defined for your default context.

Text file to add a simple user with a line and voicemail:

```
entityid|firstname|lastname|language|phonenumber|context|protocol|voicemailname|voicemailmailbox|
1|John|Doe|en_US|1000|default|sip|John Doe|1000|1234
```

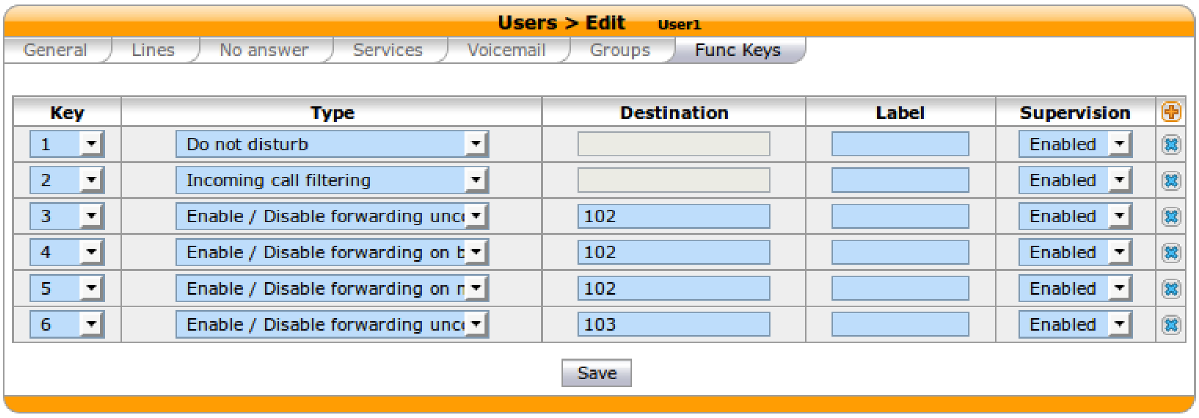
Text file to add a simple user with a line and incall:

```
entityid|firstname|lastname|phonenumber|context|protocol|incallexten|incallcontext
1|John|Doe|1000|default|sip|2050|from-extern
```

Function keys

Function keys can be configured to customize the user's phone keys. Key types are pre-defined and can be browsed through the Type drop-down list. The Supervision field allow the key to be supervised. A supervised key will light up when enabled.

Warning: SCCP device only supports type "Customized".



For User keys, start to key in the user name in destination, XiVO will try to complete with the corresponding user.

If the forward unconditionnal function key is used with no destionation the user will be prompted when the user presses the function key and the BLF will monitor *ALL* unconditionnal forward for this user.

Extensions

**3* (online call recording)

To enable online call recording, you must check the "Enable online call recording" box in the user form.

When this option is activated, the user can press `*3` during a conversation to start/stop online call recording. The recorded file will be available in the `monitor` directory of the *Services* → *IPBX* → *Audio files* menu.

**26* (call recording)

You can enable/disable the recording of all calls for a user in 2 different way:

1. By checking the "Call recording" box of the user form.

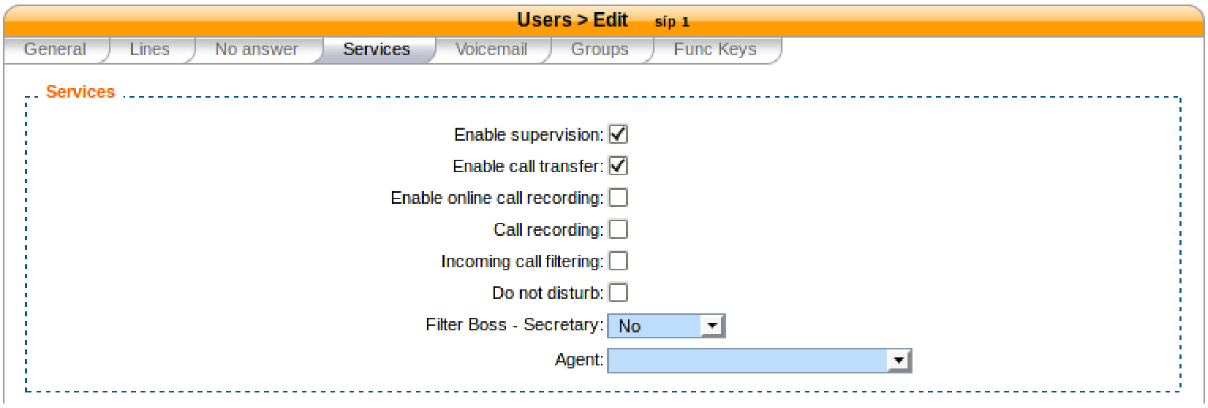


Figure 1.68: Users Services

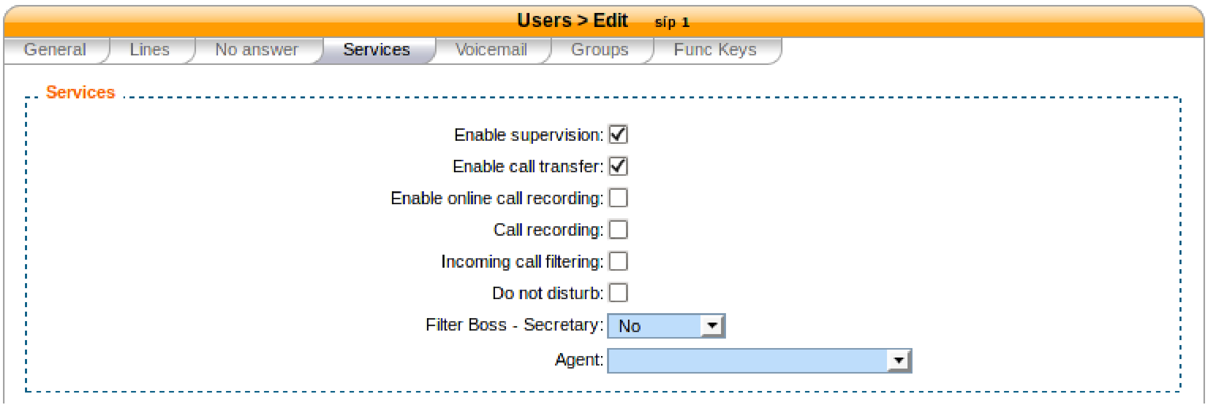


Figure 1.69: Users Services

2. By using the extension **26* from your phone (the "call recording" option must be activated in *Services* → *IPBX* → *Extensions*).

When this option is activated, all calls made to or made by the user will be recorded in the `monitor` directory of the *Services* → *IPBX* → *Audio files* menu.

1.8.29 Voicemail

Voicemail Configuration.

General Configuration

You can configure general settings for your voicemail service in *Services* → *IPBX* → *General Settings* → *Voicemails* page.

Adding voicemail

There are 2 ways to add a voicemail. First is with *Services* → *IPBX* → *IPBX settings* → *Voicemails* page, 2nd is editing user's configuration.

1 - Via *Services* → *IPBX* → *IPBX settings* → *Voicemails*

In here you can add some voicemails and configure them by clicking on the plus button.

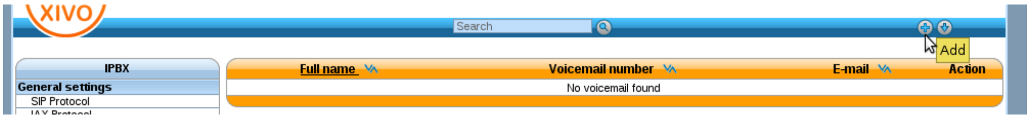


Figure 1.70: Add voicemail from voicemails menu

Once your voicemails are configured, you have to edit the users configuration to search the voicemails previously created and then associate them to your users.

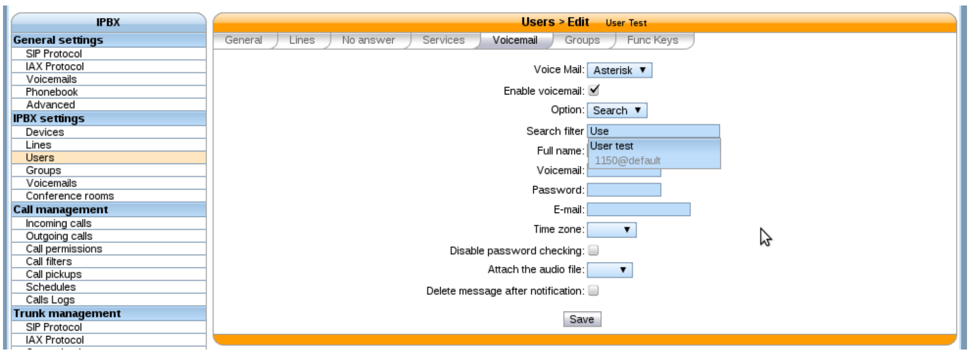


Figure 1.71: Search voicemail for specific user

2 - On user's configuration

The other way is to directly add the voicemail from user's configuration in the 'voicemail' tab :

> **Warning:** In this way, the language has to be set in user's general configuration

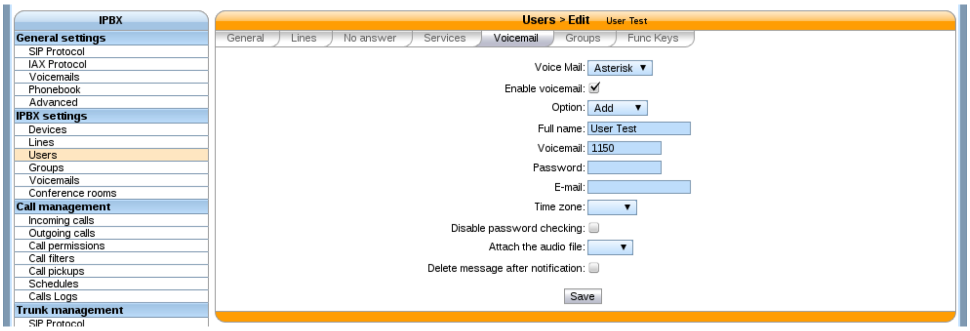


Figure 1.72: Add voicemail from user configuration

Deactivating voicemail

You can deactivate user's voicemail by un-checking 'Enable voicemail' option on the Voicemail tab from user's configuration:

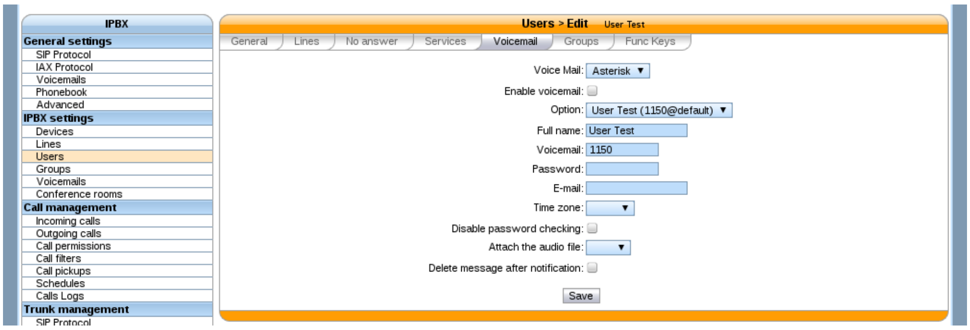


Figure 1.73: Deactivate user's voicemail

Disassociating voicemail

You can disassociate a voicemail from a user by selecting the 'None' option on 'Voice Mail' select box, from user's configuration:

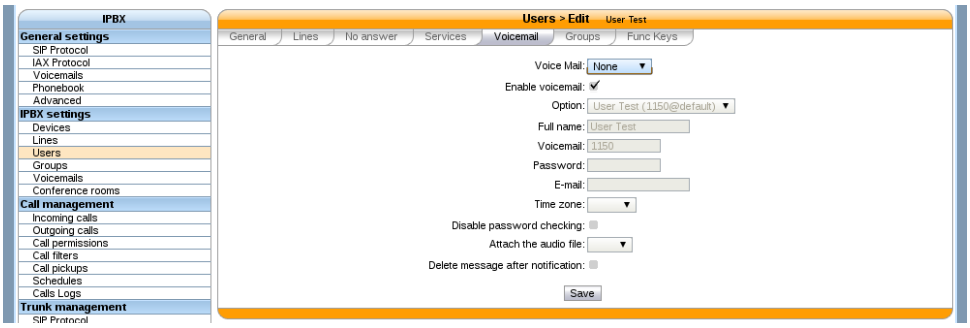


Figure 1.74: Disassociate voicemail

Warning: Note that disassociating a voicemail from its user don't delete that voicemail.

Deleting voicemail

Delete voicemail is done on *Services* → *IBX* → *IPBX settings* → *Voicemails*

Warning:

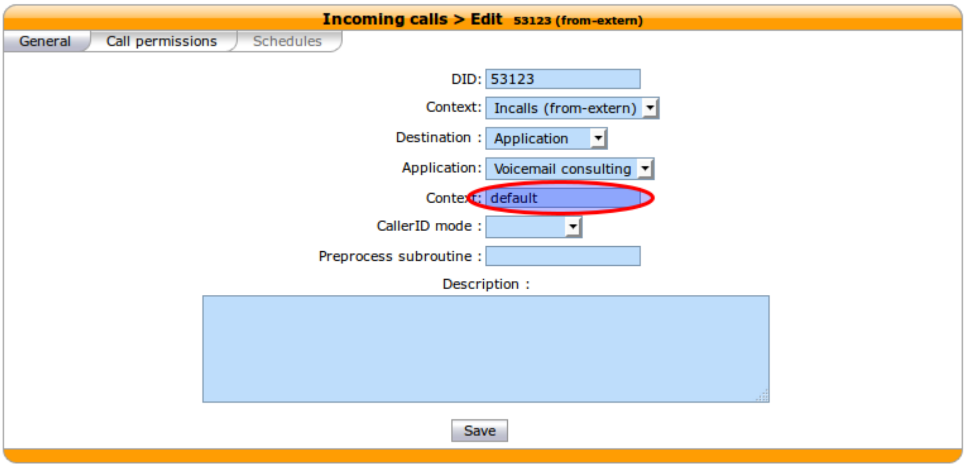
- Deleting a voicemail is irreversible. It deletes all messages associated with that voicemail.
- If concerned user still have messages waiting for him, you have to manually reboot the phone.

Disable password checking

This option allows to skip password checking for the voicemail when it is consulted from the inside. More precisely, password checking will be skipped:

- when calling the voicemail with *98
- when calling the voicemail with *99<voicemail number>

But it will not be skipped when the voicemail is consulted through an incoming call. For instance, let's consider the following incoming call:



With such a configuration, when calling this incoming call from the outside, we will be asked for:

- the voicemail number we want to consult
- the voicemail password, **even if the "Disable password checking option" is activated**

And then, we will be granted access to the voicemail.

Take note that the second "context" field contains the context of the voicemail. Voicemails of other contexts will not be accessible through this incoming call.

1.9 Contact Center

In XiVO, the contact center is implemented to fulfill the following objectives :

- Call routing

  Includes basic call distribution using call queues and skills-based routing

- Agent and Supervisor workstation.

  Provides the ability to execute contact center actions such as: agent login, agent logout and to receive real time statistics regarding contact center status

- Statistics reporting

  Provides contact center management reporting on contact center activities

- Advanced functionalities

  Call recording
- Screen Pop-up

1.9.1 Agents

Introduction

> *A call center agent is the person who handles incoming or outgoing customer calls for a business. A call center agent might handle account inquiries, customer complaints or support issues. Other names for a call center agent include customer service representative (CSR), telephone sales or service representative (TSR), attendant, associate, operator, account executive or team member.*
>
> —SearchCRM

In this respect, agents in XiVO have no fixed line and can login from any registered device.

Getting Started

- Create a user with a SIP line and a provisioned device.
- Create agents.
- Create a queue adding created agent as member of queue.

Creating agents

Service > Call center > Agents > General

These settings are specific for a given agent.

Service > Call center > Agents > Users

These settings are specific for a given agent.

Service > Call center > Agents > Queues

These settings are specific for a given agent.

Service > Call center > Agents > Advanced

These settings are specific for a given agent.

Service > IPBX > General settings > Advanced > Agent

These settings are global for all agents.

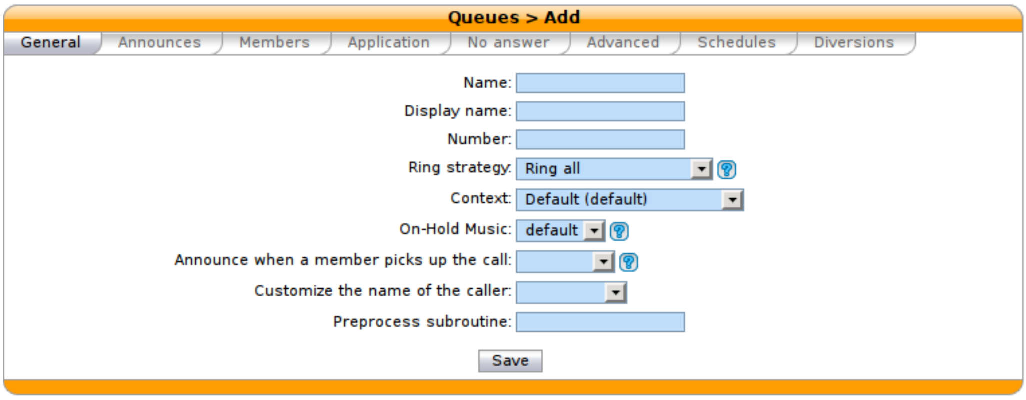


Figure 1.75: *Services* → *Call Center* → *Queues* → *Add*

1.9.2 Queues

Call queues are used to distribute calls to the agents subscribed to the queue. Queues are managed on the *Services* → *Call Center* → *Queues* page.

A queue can be configured with the following options:

- Name : used as an unique id, cannot be `general`
- Display name : Displayed on the supervisor screen

A ring strategy defines how queue members are called when a call enters the queue. A queue can use one of the following ring strategies:

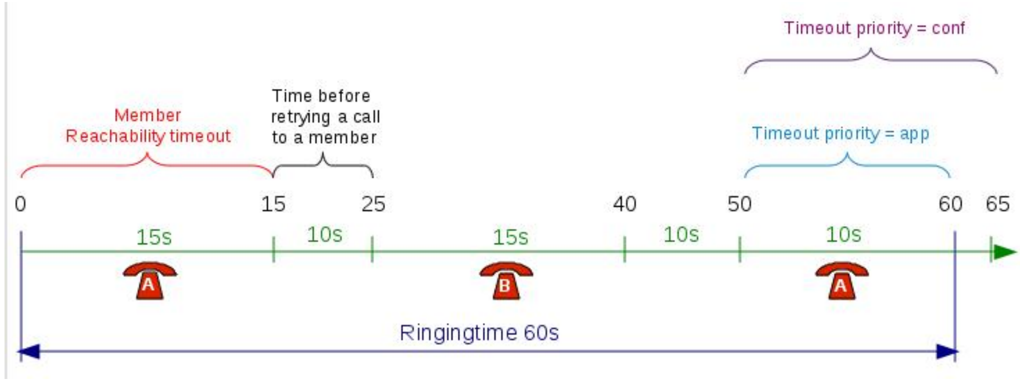
- Linear: for each call, call the first member, then the second, etc.
- Least recent: call the member who has least recently hung up a call.
- Fewest calls: call the member with the fewest completed calls.
- Round robin memory: call the "next" member after the one who answered.
- Random: call a member at random
- Weight random: same as random, but taking the member penalty into account.
- Ring all: call all members at the same time.

Warning: When editing a queue, you can't change the ring strategy to linear. This is due to an asterisk limitation. Unfortunately, if you want to change the ring strategy of a queue to linear, you'll have to delete and create a new queue with the right strategy.

Timers

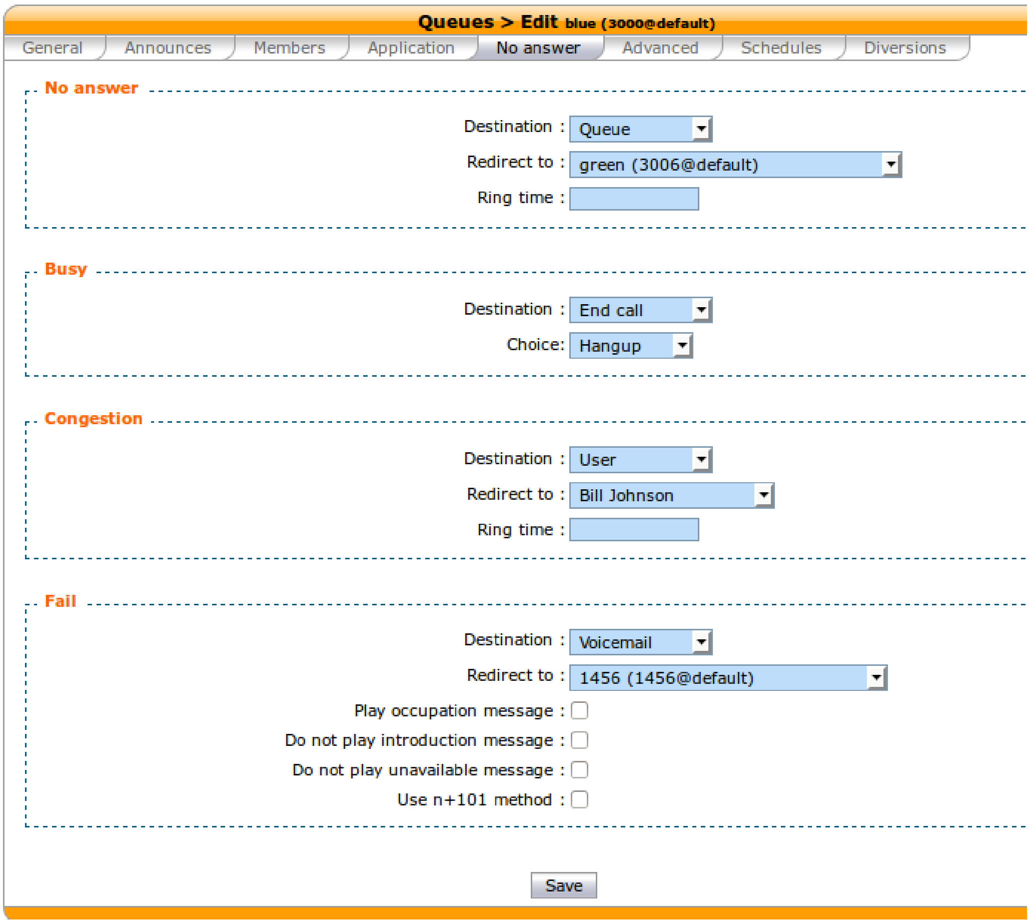
You may control how long a call will stay in a queue using different timers:

- Member reachabillity time out (Advanced tab): Maximum number of seconds a call will ring on an agent's phone. If a call is not answered within this time, the call will be forwareded to another agent.
- Time before retrying a call to a member (Advanced tab) : Used once a call has reached the "Member reachability time out". The call will be put on hold for the number of seconds alloted before being redirected to another agent.
- Ringing time (Application tab) : The total time the call will stay in the queue
- Timeout priority (Application tab) : Determines which timeout to use before ending a call. When set to "configuration", the call will use the "Member reachability time out". When set to "dialplan", the call will use the "Ringing time".



No Answer

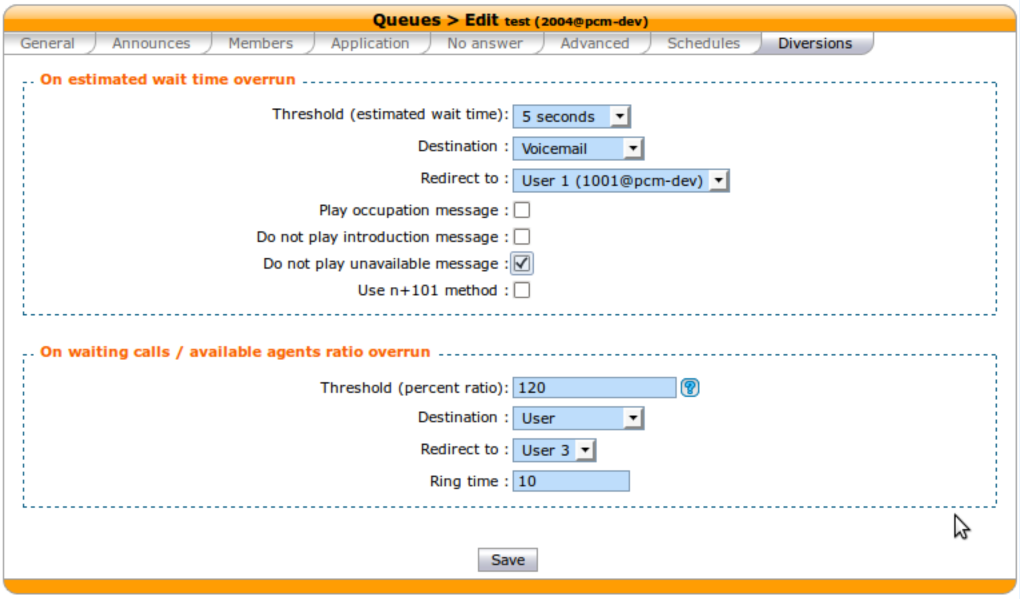
Call can be diverted on no answer :



- No answer : The call reach the "Ringing time" in Application tab and no agent has answered the call
- Congestion : The number of calls waiting have reach the "Maximum number of people allowed to wait:" limit of advanced tab
- Fail : No agent was available to answer the call when call entered the queue (join an empty queue condition advanced tab) or the call was queued and no agents was available to answer (Remove callers if there are no agents advanced tab)

Diversions

Diversions can be used to redirect calls towards another destination when a queue is very busy. Calls are redirected using one of the two following scenarios:



Estimated Wait Time Overrun

When this scenario is used, the administrator can set a destination for calls when the average waiting time is over the threshold.

Waiting Calls / Available Agents Ratio

When this scenario is used, the administrator can set a destination when the call ratio is higher than the percent threshold. The call ratio is calculated with the following formula:

```
call ratio = (number of waiting calls / available agents) * 100
```

Here are a few examples:

```
Threshold: 100%
Waiting calls: 3
Available agents: 2
call ratio = (3 / 2) * 100 = 150%
Calls will be redirected


Threshold: 120%
Waiting calls: 9
Available agents: 12
call ratio = (9 / 12) * 100 = 75%
Calls will not be redirected
```

> **Warning:** With a threshold under 100% and only one agent logged, no call will distributed since one waiting call / one agent = 100%

1.9.3 Supervision

Introduction

Allows a contact center supervisor to monitor contact center activities such as:

- Monitoring real time information from call queues
- Agent activities per call queues
- Agent detailed activities

XiVO client as a Supervision Platform

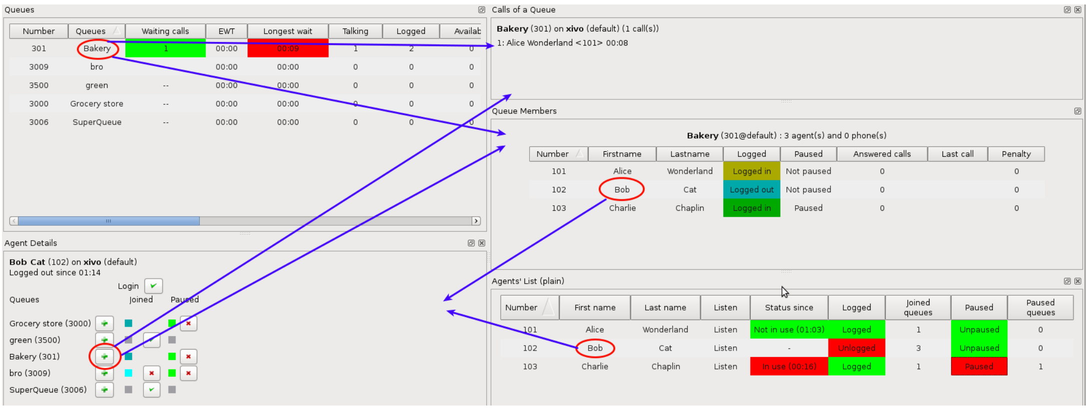
Configuration

A supervisor profile defined in *Service* → *CTI Server* → *Profiles* menu usually contains the following Xlets :

- Identity
- Queues
- Queue members
- Queues (entries detail)
- Agents (list)
- Agents (detail)

  Note You may also see the *Agent Status Dashboard*

Supervision Panel

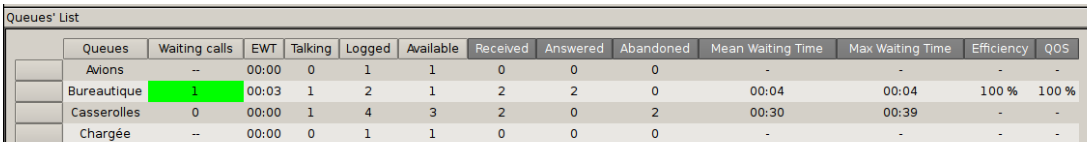


- Clicking on a queue's name in the queue list will display the agent list in the xlet *Queue Members* and show waiting calls in the *Calls of a Queue* xlet.
- Clicking on an agent's name in the agent list will display information on the agent in the *Agent Details* xlet
- Clicking on the + icon in the *Agent Details* xlet will display information about the selected queue in the *Calls of a Queue* and *Queue Members* xlets.

Queue List General information

The queue list is a dashboard displaying queue statistics and real-time counters for each queue configured on the XiVO.

Columns



Queues' List

Queues	Waiting calls	EWT	Talking	Logged	Available	Received	Answered	Abandoned	Mean Waiting Time	Max Waiting Time	Efficiency	QOS
Avions	--	00:00	0	1	1	0	0	0	-	-	-	-
Bureautique	1	00:03	1	2	1	2	2	0	00:04	00:04	100 %	100 %
Casserolles	0	00:00	1	4	3	2	0	2	00:30	00:39	-	-
Chargée	--	00:00	0	1	1	0	0	0	-	-	-	-

Queues queue name and number if configured to be displayed

Waiting calls The number of calls currently waiting for an agent in this queue. The background color can change depending of the configured thresholds

EWT Estimated waiting time

Longest wait The longest waiting time for currently waiting calls. The background color can change depending of the configured thresholds

Talking The number of agents currently in conversation in the queue. This column is set to 0 when the queue has just been created and no members have been added.

Logged The number of logged agents in the queue. This column is set to "N/A" when the queue has just been created and no members have been added.

Available The number of available agents ready to take a call in the queue. This column is set to N/A when the queue has just been created and no members have been added.

Received The number of calls received in this queue during the configured statistical window

Answered The number of calls answered in this queue during the configured statistical window

Abandoned The number of calls abandoned in this queue during the configured statistical window

Mean waiting time The mean wait time in the statistical time window, in mm:ss If no calls are received, "-" is displayed

Max waiting time The longest wait time in the statistical time window, in mm:ss If no calls are received, "-" is displayed

Efficiency Answered calls over received calls during the configured statistical window (unanswered calls that are still waiting are not taken into account). If no calls are received, "-" is displayed

QOS Percentage of calls taken within X seconds over answered calls during the configured statistical window. If no calls are received, "-" is displayed

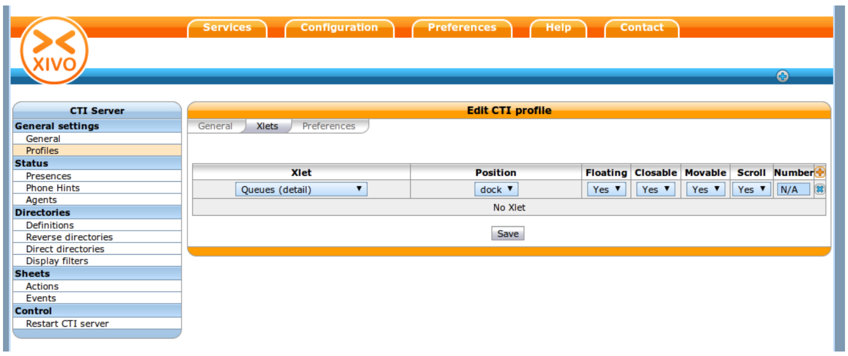
Counter availability

When the XiVO client is started, "na" is diplayed for counters that have not been initialized.

When the XiVO client is restarted, the counters are always displayed and calculated as if the application was not restarted. When the server is restarted, counters are reinitialized.

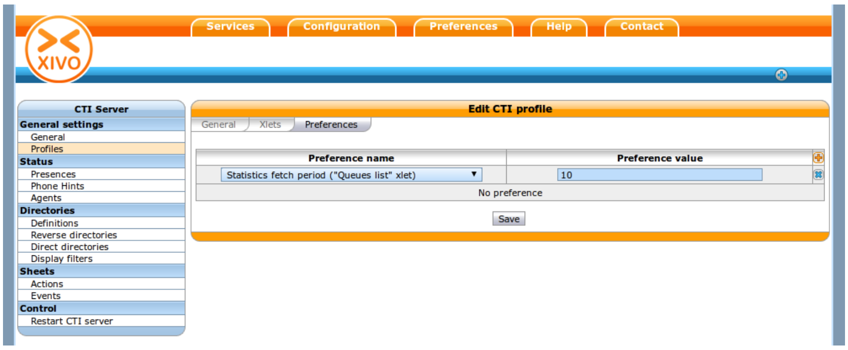
Enabling the xlet

The xlet can be added to any CTI profile from the web interface.



Configuration

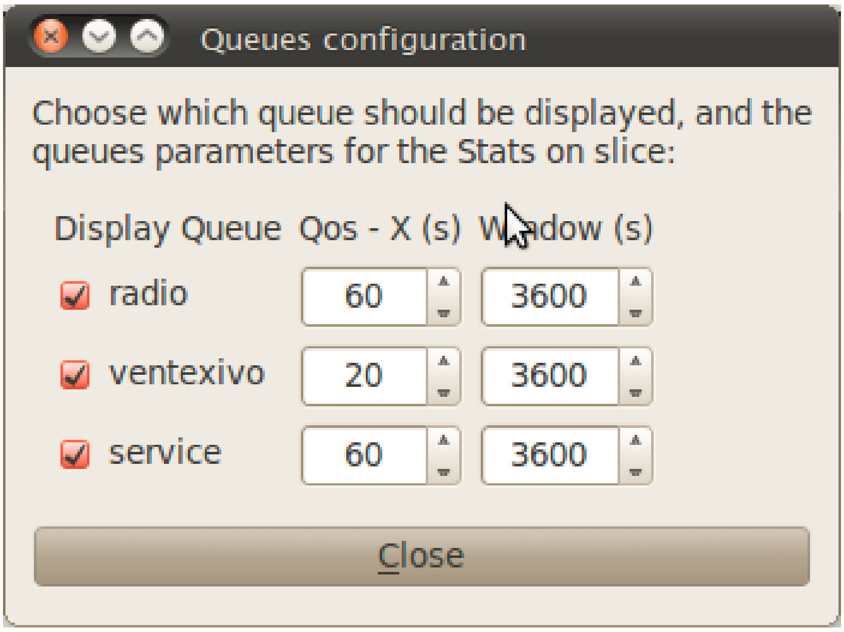
Some values can be configured for the xlet. The statistic fetch timer can be set in the CTI profile preferences. This option is expressed in seconds and the default is 30 seconds.



The statistical period can be configured through the XiVO client once logged in by right-clicking on the Queue's name in the *Queues* xlet. For each queue, you can configure the following information:

- Qos: maximum wait time for a call, in seconds.
- Window: period of time used for accumulating statistics, in seconds.

The data used to compute statistics on the XiVO server is only kept for a maximum of 3 hours. The window period cannot be configured to go beyond this limit.



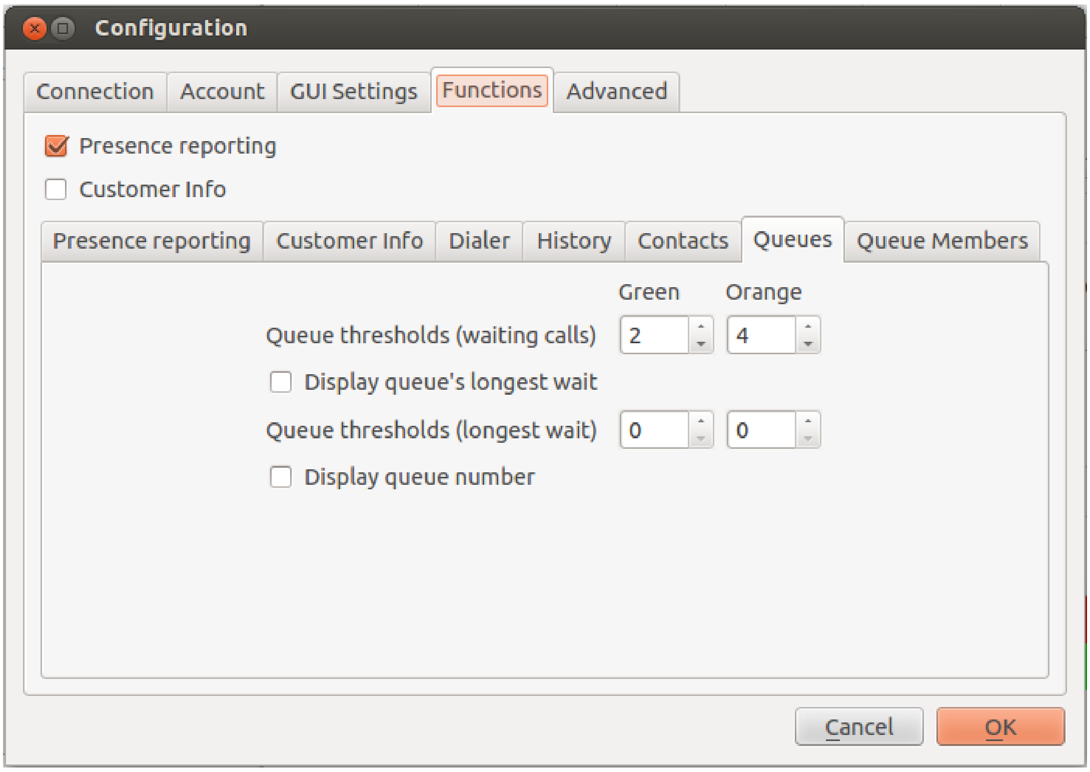
Display options can also be set on the client side. A threshold can be configured to change the color of a column using the following parameters:

- Queue thresholds (waiting calls): number of waiting calls in the queue.
- Display queue's longest wait: Add a column displaying the number of seconds the longest call has waited.
- Queue thresholds (longest wait): number of seconds for the longest waiting call in the queue.
- Display queue number: Add a column displaying the queue's number.

Monitoring queues on high dimension screens

You may want to display the queue list on one big screen, visible by multiple people. However, the default font will not be large enough, so the information will not be readable.

You can change the font size of this Xlet by giving a configuration file when launching the XiVO Client:




```
> xivoclient -stylesheet big_fonts.qss   # Windows and Mac
$ xivoclient -- -stylesheet big_fonts.qss   # GNU/Linux
```

The `big_fonts.qss` file should contain:

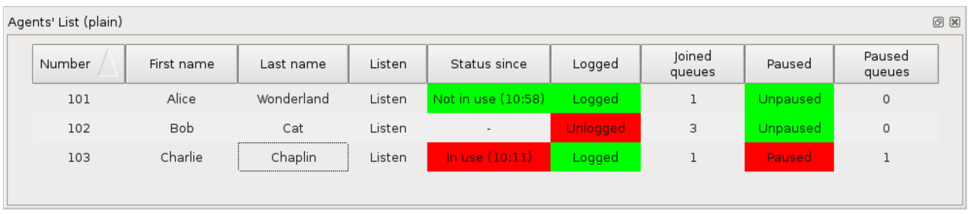
```
QueuesView {font-size: 40px;}
QueuesView QHeaderView {font-size: 40px;}
```

Units of size that can be used are described on the Qt documentation.

Agent List

General information

The queue list is a dashboard displaying each agent configured on the XiVO.



Columns

Number The agent's number

First name & Last name The agent's first name and last name

Listen A *clickable cell* to listen to the agent's current call.

Clicking on the cell will make your phone ring. When you'll answer, you'll hear the conversation the agent is having.

You'll then be able to press the following digits on your phone to switch between the different "listen" modes:

- 4 - spy mode (default). No one hears you.
- 5 - whisper mode. Only the agent hears you.
- 6 - barge mode. Both the agent and the person he's talking to hear you.

Status since Shows the agent's status and the time spent in this status. An agent can have three statuses:

- *Not in use* when he is ready to answer an ACD call
- *Out of queue* when he called or answered a call not from the queue
- *In use* when he is either on call from a queue, on pause or on wrapup

Logged A *clickable cell* to log or unlog the agent

Joined queues The number of queues the agent will be receiving calls from

Paused A *clickable cell* to pause or unpause the agent

Paused queues The number of queues in which the agent is paused

Agent Details **General information**

Display advanced informations of an agent and enable to login/logoff, add/remove to a queue, and pause/unpause.

1. This is the status information of agent
2. Button to login/logoff agent
3. Supervision button of the Xlet "Calls of a queue"
4. Pause/Unpause button for given queue
5. Add/Remove agent for given queue

Queue members The queue members lists which agents or phones will receive calls from the selected queue and some of their attributes.

Columns

Number The agent number or the phone number of the queue member.

Firstname and Lastname First name and last name of the agent or the user to which the phone belongs.

Logged Whether the agent is logged or not. Blank for a phone.

Paused Whether the agent is paused or not. Blank for a phone.

Answered calls Number of calls answered by the member since last login (for an agent), or restart or configuration reload.

Last call Hangup time of the last answered calls.

Penalty Penalty of the queue member.

Link XiVO Client presence to agent presence

You can configure XiVO to have the following scenario:

- The agent person leaves temporarily his office (lunch, break, ...)
- He sets his presence in the XiVO Client to the according state
- The agent will be automatically set in pause and his phone will not ring from queues
- He comes back to his office and set his presence to 'Available'
- The pause will be automatically cancelled

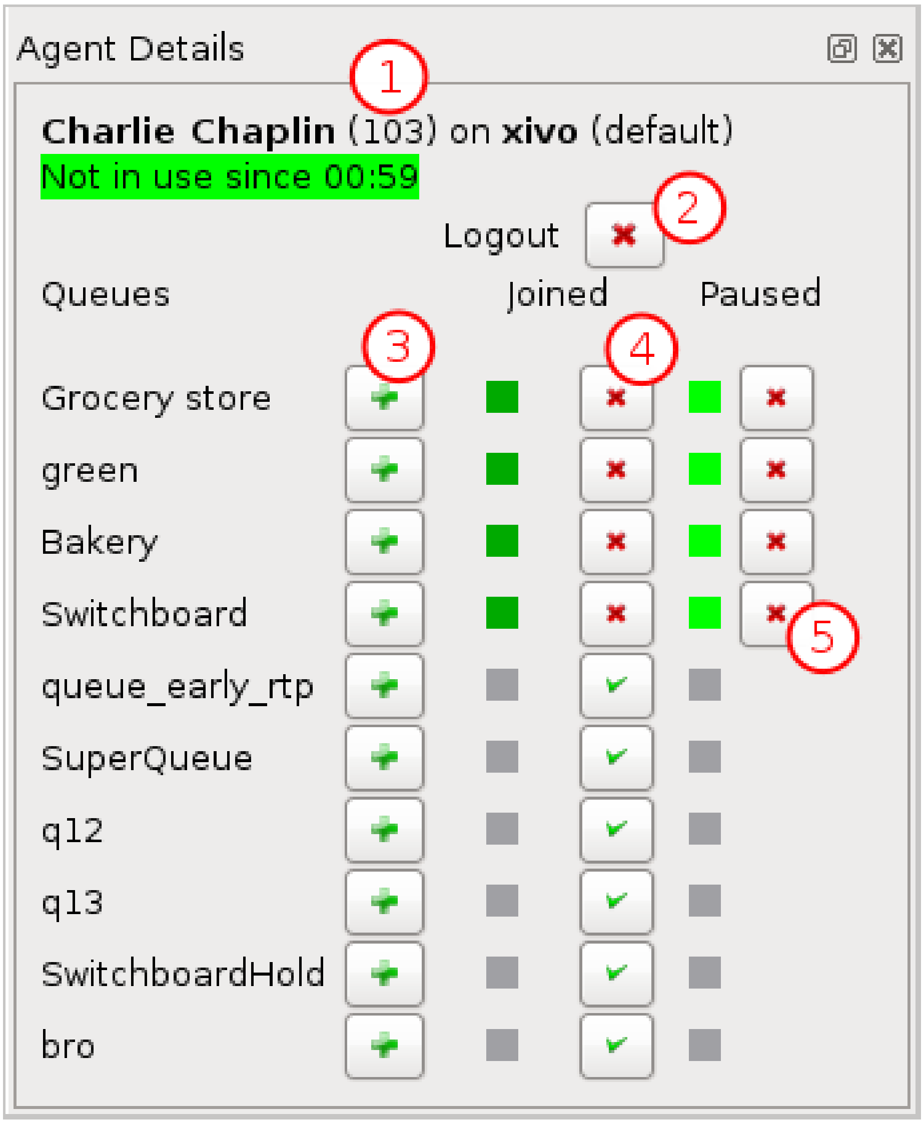
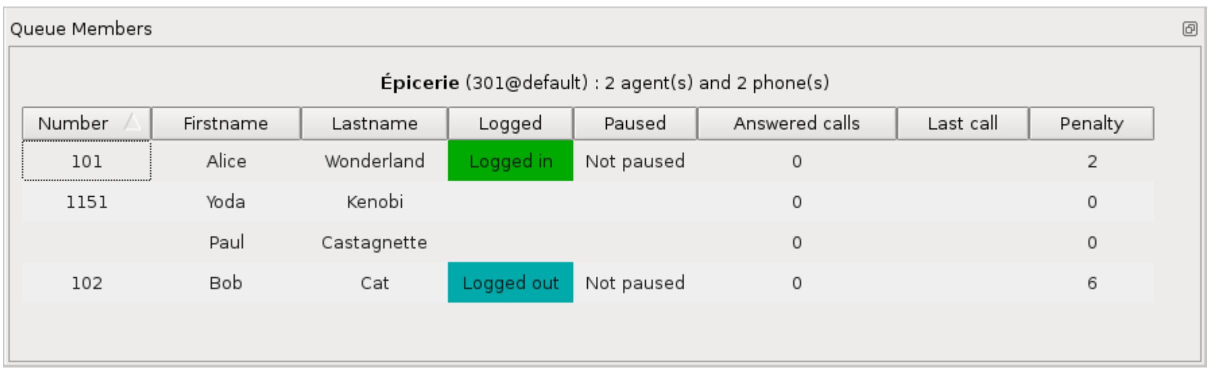


Figure 1.76: Agent Details



You can *configure the presence states* of CTI profiles and attach `Actions` to them, such as *Set in pause* or *Enable DND*.

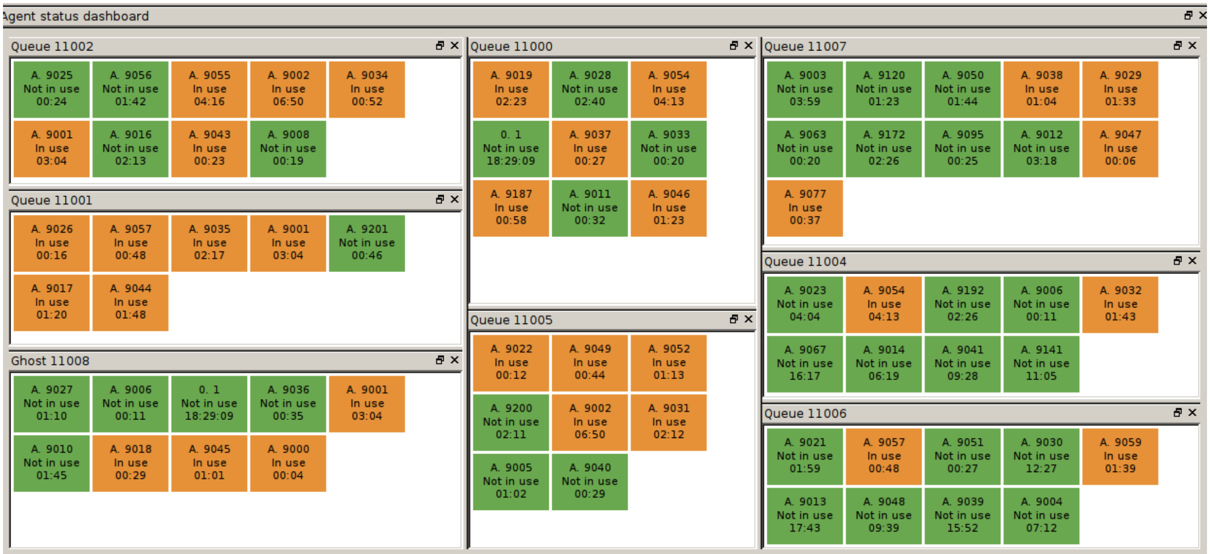
You can then attach an action *Set in pause* for multiple presence states and attach an action *Cancel the pause* for the presence state *Available*.

For now, the actions attached to the mandatory presence *Disconnected* will not be taken into account.

1.9.4 Agent Status Dashboard

Overview

The goal of the agent status dashboard xlet is to give contact center supervisors a better overview of agent status evolution in active queues.



Usage

The xlet is *read-only* and presents a list of queues. For each queue, the xlet displays a status box for each logged in agent. Each status box gives the following information:

- Agent name
- Agent status: Shows the agent's status. An agent can have six statuses:
  - *Not in use* when he is ready to answer an ACD call
  - *Int. Incoming* when he answered an internal call not from a queue
  - *Int. Outgoing* when he emitted an internal call not from a queue
  - *Ext. Incoming* when he answered an external call not from a queue
  - *Ext. Outgoing* when he emitted an external call not from a queue
  - *In use* when he is either on call a from a queue, on pause or on wrapup
- Agent status since: Shows the time spent in the current status
- Background color:
  - green if *Not in use*
  - purple if *Int. Incoming* or *Int. Outgoing*
  - pink if *Ext. Incoming* or *Ext. Outgoing*

   - orange if *In use*

Note that the agent status will only change when the communication is established, not when phones are ringing.

Known bugs

1. If an agent emits a call via his XiVO Client, the status will change to *Int. Outgoing* or *Ext. Outgoing* when the destination phone rings, instead of when the destination answers.
2.
   - Given the agent is on an ACD call
   - When the agent logs out
   - When the agent hangs up the ACD call
   - When the agent logs back in via CTI Client
   - Then the agent may be seen as outgoing non-ACD communication, whether there is a non-ACD communication or not

   To make the agent *Not in use* again, make a non-ACD call and hangup.
3.
   - Given the agent is on ACD call
   - When the agent calls someone else (e.g. his supervisor)
   - When the ACD call hangs up (while the agent talks to his supervisor)
   - Then the agent is seen as available, instead of in outgoing non-ACD communication.

   This applies to all kinds of non-ACD calls.

Changing the disposition

The disposition of the Xlet can be changed in two ways:

- Placement of queues
- Which queues are displayed

The disposition is saved whenever the XiVO Client is closed and restored when it is opened again.

Changing the placement of queues

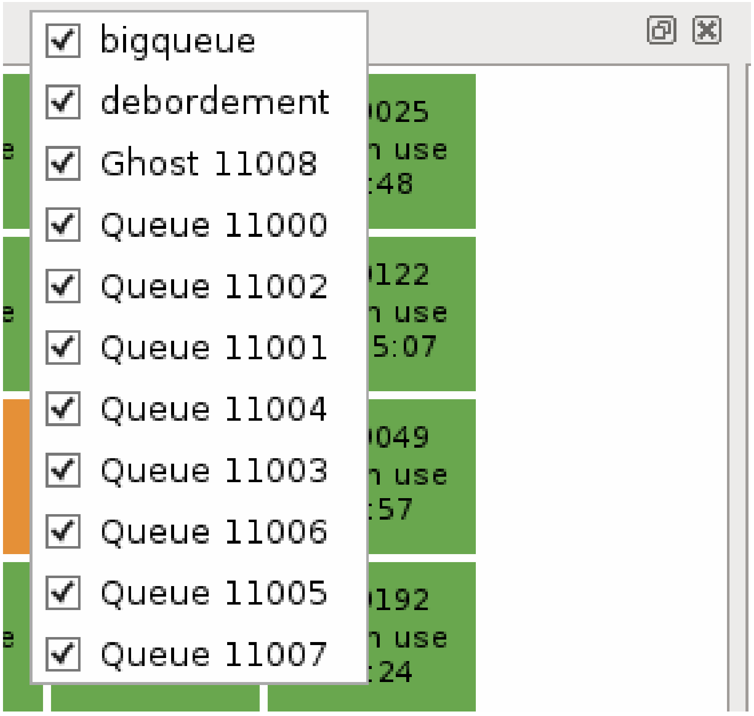
The little windows containing each queue can be resized and moved around. That way, any layout can be achieved, according to the size and importance of each queue.

Choosing which queues are displayed

There is a little contextual menu when right-clicking on the title bar of every queue window. Checking/unchecking the lines of this menu shows/hides the associated queue.

Known issues

- There is no profile containing this xlet. The profile must be created manually.
- There is no sorting on agents in a queue.
- An empty queue will display an empty box with no message specifying the queue has no logged agents.



Configuration

No special configuration is necessary other than creating a CTI profile in which the Agent Status Dashboard is added.

1.9.5 Skills-Based Routing

Introduction

> *Skills-based routing (SBR), or Skills-based call routing, is a call-assignment strategy used in call centres to assign incoming calls to the most suitable agent, instead of simply choosing the next available agent. It is an enhancement to the Automatic Call Distributor (ACD) systems found in most call centres. The need for skills-based routing has arisen, as call centres have become larger and dealt with a wider variety of call types.*
>
> —Wikipedia

In this respect, skills-based routing is also based on call distribution to agents through waiting queues, but one or many skills can be assigned to each agent, and call can be distributed to the most suitable agent.

In skills-based routing, you will have to find a way to be able to tag the call for a specific skill need. This can be done for example by entering the call distribution system using different incoming call numbers, using an IVR to let the caller do his own choice, or by requesting to the information system database the customer profile.

Getting Started

- Create the skills
- Apply the skills to the agents
- Create a skill rule sets
- Assign the skill rule sets using a configuration file
- Apply the skill rule sets to call qualification, i.e. incoming calls by using the preprocess routine field.

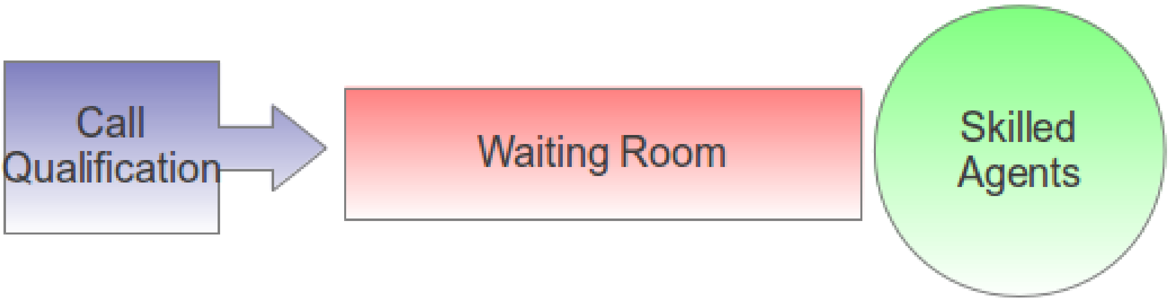


Figure 1.77: Skills-Based Routing

Skills

Skills are created using the menu *Services* → *Call center* → *Skills*. Each skill belongs to a category. First create the category, and in this category create different skills. Note that the skill names can't contain upper case letters.

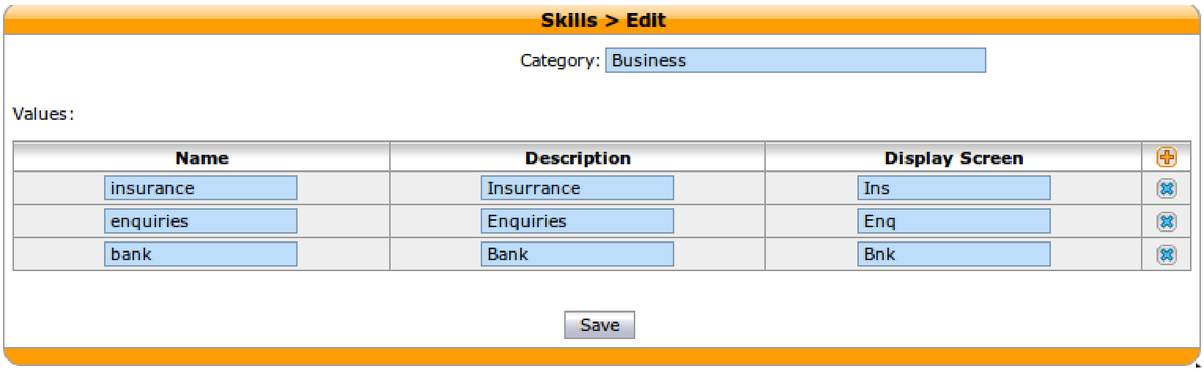


Figure 1.78: Skills Creation

Once all the skills are created you may apply them to the agents. Agents may have one or more skills from different categories.

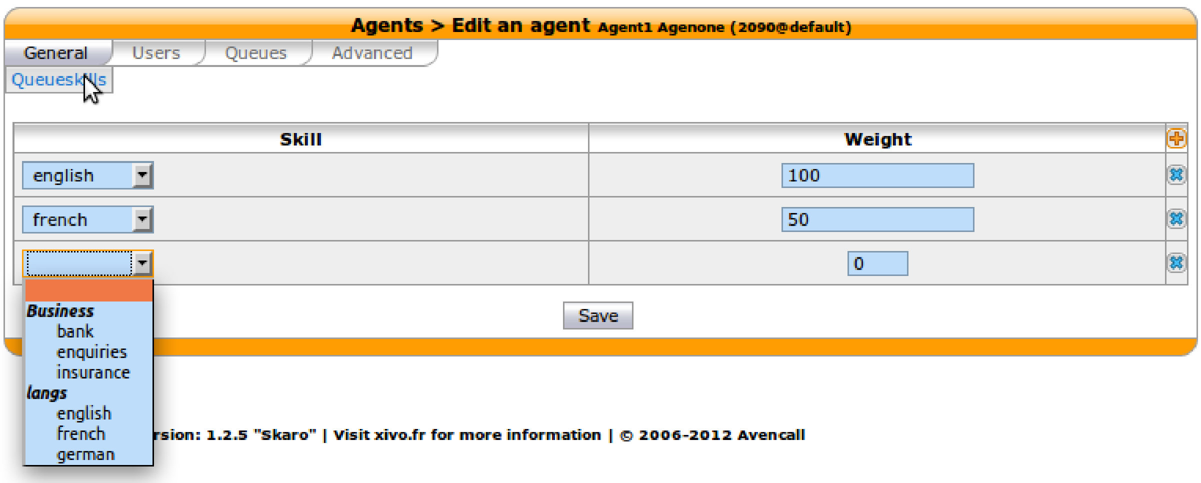


Figure 1.79: Apply Skills to Agents

Skill Rule Sets

Once skills are created, rule sets can be defined.

Rules are the way to reach the right agent. Rules can be composed and dynamically modified

A rule set is a list of rules. Rules are evaluated against each queue member (agent) in order to see if it matches. The call is distributed according to the matching rule.

Each rule has two parts:

- the first part is a dynamical condition. If its evaluation is false, the next rule is tried;
- the second part is tested against queue member's skills, to define a selection.

Operators

Arithmetic and logical operators can be applied to rules :

- operand1 / operand2 (division)
- operand1 * operand2 (multiplication)
- operand1 - operand2 (substraction)
- operand1 + operand2 (addition)
- operand1 ! operand2 (is not equal)
- operand1 = operand2 (is equal)
- operand1 > operand2 (is greater than)
- operand1 < operand2 (is lesser than)
- operand1 & operand2 (both are true)
- operand1 | operand2 (at least one of them are true)

  '/' is the operator with the higher priority, and '|' the one with the lower priority. You can use brackets '()' to overload operator priorities.

Dynamical Part

The first part is evaluated after a selection of queue members is created with the rules from the second part. The result of this evaluation will determine if this rule can be kept or if the selection is to be done with the next rule. This can be understood as an `if` statement :

```
if this condition is true then
   select agents with skills (evaluate second part)
else
   evaluate next rule
```

On this part, these variables can be used :

> **Warning:** you need two persons waiting in the queue for these variables be taken into account

- EWT (Estimated Waiting Time) The waiting time estimated for the current selection of members
- WT (Waiting time) The time that caller has been waiting

  Example

  WT < 60, french = 100
  french > 50

  If the waiting time is less than 60 seconds, select an agent speaking good french (100), otherwise select an agent with low level of french

Skill Part

This second part is evaluated against every queue member's skills, to know if it is selected or not.

Variables are skills names, which you can check with operators above. You can also use meta-variables, starting with a '$', to substitute them with data set on the Queue() call. For example, if you call Queue() with the skill rule set argument equal to:

```
select_lang(lang=german)
```

Then every `$lang` occurrence will be replaced by 'german'.

Figure 1.80: Create Skill Rule Sets

Examples

```
[tech]
rule => WT < 60, technic & ($os > 29 & $lang > 39 | $os > 39 & $lang > 19)
rule => WT < 120, technic & ($os > 19 & $lang > 39 | $os > 29 & $lang > 19)
rule => WT < 3600, technic & $os > 10 & $lang > 19
rule => technic

[client-standard]
rule => technic = 0 & (sympathy > 20 | linux > 10 & windows > 10)

[client-request]
rule => EWT < 120, technic = 0 & (sympathy > 60)
rule => technic = 0
```

Apply Skill Rules

Once skills, skill rules are created, they can attached to the call using a bit of dialplan. This dialplan is stored in a configuration file you may edit using menu *Services* → *IPBX* → *Configuration Files*.

In the figure above, 3 different languages are selected using three different subroutines.

Note: Do not forget to issue a dialplan reload in Asterisk CLI after configuration file modification.

Each of this different selections of subroutines can be applied to the call qualifying object. In the following example language selection is applied to incoming calls.

Example

Configuration file for simple skill selection :

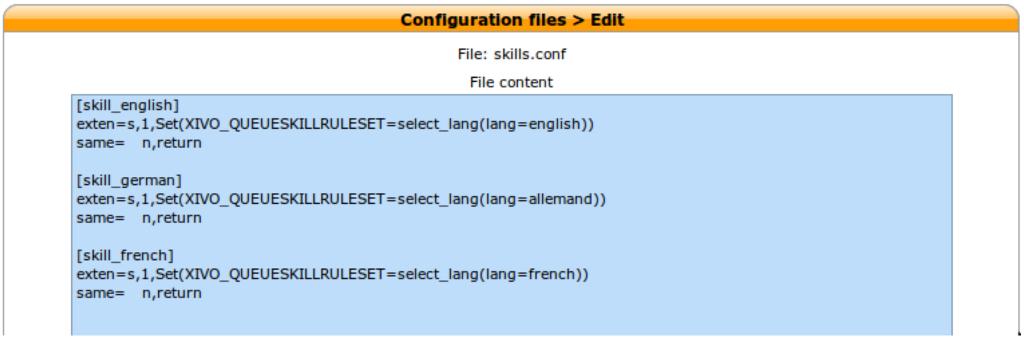


Figure 1.81: Use Rule Set In Dialplan

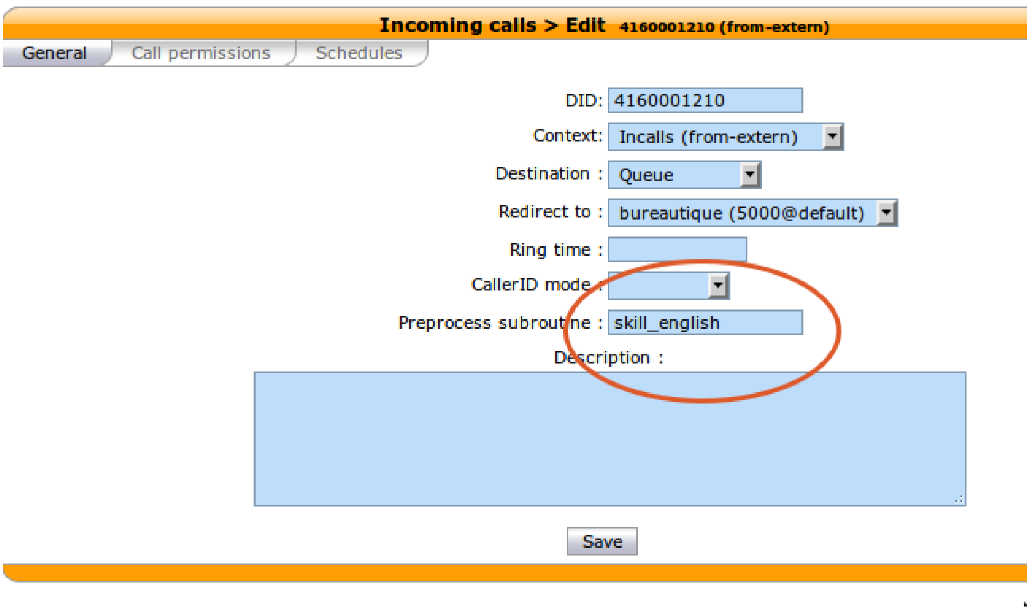


Figure 1.82: Apply Rule Set to Incoming Call

```
[simple_skill_english]
exten = s,1,Set(XIVO_QUEUESKILLRULESET=english_rule_set)
same  =   n,Return()

[simple_skill_french]
exten = s,1,Set(XIVO_QUEUESKILLRULESET=french_rule_set)
same  =   n,Return()
```

In this example you just need to create two simple skill rule sets, one named english_rule_set with a rule english > 90 and the other named french_rule_set

Monitoring

You may monitor your waiting calls with skills using the asterisk CLI and the command `queue show <queue_name>`:

```
xivo-jylebleu*CLI> queue show services
services has 1 calls (max unlimited) in 'ringall' strategy (0s holdtime, 2s talktime), W:0, C:1, A
  Members:
     Agent/2000 (Not in use) (skills: agent-1) has taken no calls yet
     Agent/2001 (Unavailable) (skills: agent-4) has taken no calls yet
  Virtual queue english:
  Virtual queue french:
     1. SIP/jyl-dev-assur-00000017 (wait: 0:05, prio: 0)
  Callers:
```

You may monitor your skills groups with the command `queue show skills groups <agent_name>`:

```
xivo-jylebleu*CLI> queue show skills groups <PRESS TAB>
agent-2    agent-3    agent-4    agent-48   agent-7    agent-1
xivo-jylebleu*CLI> queue show skills groups agent-1
Skill group 'agent-1':
  - bank              : 50
  - english           : 100
```

You may monitor your skills rules with the command `queue show skills rules <rule_name>`:

```
xivo-jylebleu*CLI> queue show skills rules <PRESS TAB>
english      french       select_lang
xivo-jylebleu*CLI> queue show skills rules english
Skill rules 'english':
  => english>90
```

1.9.6 Statistics

Overview

The statistics page is used to monitor the efficiency of queues and agents. Statistics are automatically generated every six hours. They can also be generated manually.

Configuration

In order to display call center statistics, you must create at least one configuration profile.

The configuration profile is used to generate reports from the cache. The cache is generated independently from the configuration so adding a new configuration does not require a new cache generation.

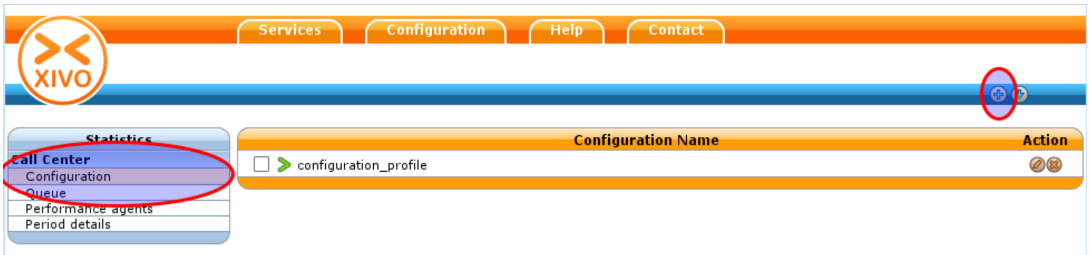


Figure 1.83: Statistics Configuration

Configuration options

Field	Values	Description
name	string	Configuration name, useful for remembering what the configuration is used for
interval	enum [0-999] [day, week, month]	Default time interval used when displaying statistics. Examples: "-1 day": show statistics for yesterday "-3 weeks": show statistics for the last 3 weeks
show on summary page		Display this configuration on the summary page
timezone	America/Montreal	Your time zone
Period cache		Maximum and minimum dates that can be used for displaying statistics
start	YYYY-MM	Start date
end	YYYY-MM	End date. If left to 0, use the servers' current date
Working Hours		Work hours for agents
start	hh:mm	Beginning of working hours.
end	hh:mm	End of working hours
Periods		Number of calls answered for a time period
Period 1	number of seconds (Example: 20)	Show number of calls answered within 20 seconds in column "P1"
Period n	number of seconds (Example: 20)	Show number of calls answered within 20 seconds in column "Pn"

Note: Calls outside of working hours will not be in the cache. e.g. if working hours are from 8:00 AM to 16:00 PM, a call at 7:55 AM will not show up in the reports.

Note: Statistics are computed on the hour. e.g. If work hours are from 8:30 to 16:15, working hours should be set from 8:00 to 17:00.

Note: Period includes both bounds, if the same number is used for the higher bound and the lower bound of next period, some calls will be counted twice. i.e period 1 : 0-30 period 2 : 31-60 period 3 : 61

How to generate the cache

The cache must be generated before using reports. By default, the cache is automatically generated every six hours.

However, you can safely generate it manually. The script to generate the cache is *xivo-stat fill_db*. When this script is run, statistics will be regenerated for the last 8 hours starting from the previous hour. e.g. If you run xivo-stat on 2012-08-04 11:47:00, statistics will be regenerated from 2012-08-04 03:00:00 to 2012-08-04 11:47:00

Note: *xivo-stat fill_db* can be a long operation when used for the first time or after a *xivo-stat clean_db*

Warning: The current events have an end date of the launch date of the script xivo-stat as the end date.

Clearing the cache

If for some reason the cache generation fails or the cache becomes unusable, the administrator can safely clean the cache using *xivo-stat clean_db* and then regenerate it. This operation will only clear the cache and does *not* erase any other data.

Queue statistics

Queue statistics can be viewed in *Services* → *Statistics* → *Queue*.

The first table displays a list of queues with all the counters for the period choosen from the Dashboard panel

				Dissuaded or Overflowed						
	Received	Answered	Abandoned	Closed	Saturated	NA	Blocking	AWT	Answered rate	QoS
STA1	1749	1521	84	21	0	0	123	00:00:13	88 %	0 %
STA2	1713	1454	73	38	0	0	148	00:00:11	86 %	0 %
STA3	1529	1367	76	23	0	0	63	00:00:10	90 %	0 %
STA4	2147	1776	115	17	0	0	239	00:00:16	83 %	0 %
STA5	1800	1594	93	28	0	0	85	00:00:13	89 %	0 %

By clicking on a queue name you may display detailed queue statistics

Statistics can be displayed :

By week

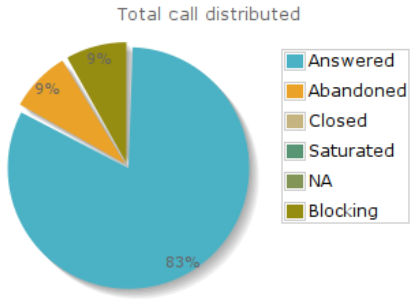
By month

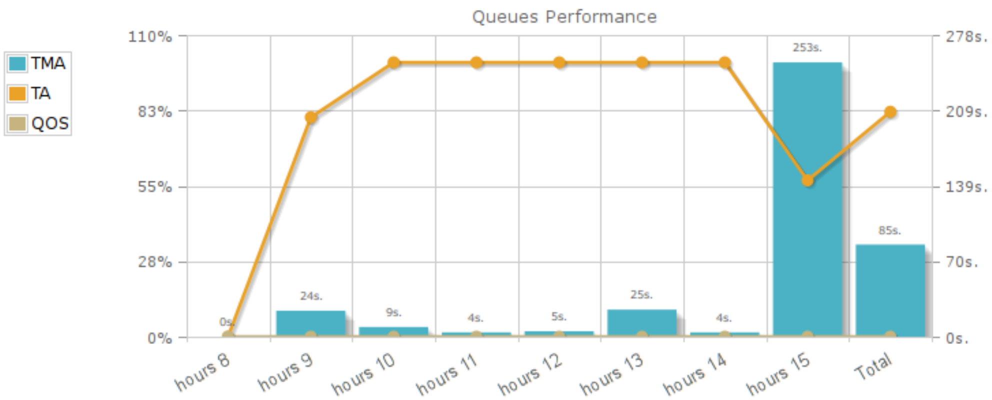
By year

Counters

- Received: Total number of received calls
- Answered: Calls answered by an agent
- Abandoned: Calls that were hung up while waiting for an answer
- Dissuaded or Overflowed:
  - Closed: Calls received when the queue was closed
  - No answer (NA): Calls that reached the ring timeout delay
  - Satured: Calls received when the queue was already full (“Maximum number of people allowed to wait:” limit of advanced tab) or when one of the diversion parameters were reached

				Dissuaded or Overflowed						
	Received	Answered	Abandoned	Closed	Saturated	NA	Blocking	AWT	Answered rate	QoS
8h-9h	0	0	0	0	0	0	0			
9h-10h	5	4	1	0	0	0	0	00:00:24	80 %	0 %
10h-11h	2	2	0	0	0	0	0	00:00:09	100 %	0 %
11h-12h	1	1	0	0	0	0	0	00:00:04	100 %	0 %
12h-13h	3	3	0	0	0	0	0	00:00:05	100 %	0 %
13h-14h	1	1	0	0	0	0	0	00:00:25	100 %	0 %
14h-15h	4	4	0	0	0	0	0	00:00:04	100 %	0 %
15h-16h	7	4	1	0	0	0	2	00:04:13	57 %	0 %
Total	23	19	2	0	0	0	2	00:01:25	82 %	0 %

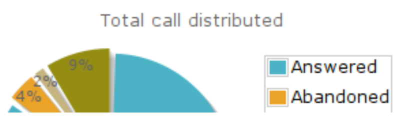




				Dissuaded or Overflowed						
	Received	Answered	Abandoned	Closed	Saturated	NA	Blocking	AWT	Answered rate	QoS
Monday 7	28	27	0	0	0	0	1	00:00:10	96 %	0 %
Tuesday 8	22	20	1	0	0	0	1	00:00:04	90 %	0 %
Wednesday 9	23	19	2	0	0	0	2	00:01:25	82 %	0 %
Thursday 10	21	21	0	0	0	0	0	00:00:07	100 %	0 %
Friday 11	34	30	1	0	0	0	3	00:00:08	88 %	0 %
Total	128	117	4	0	0	0	7	00:00:21	91 %	0 %

				Dissuaded or Overflowed						
	Received	Answered	Abandoned	Closed	Saturated	NA	Blocking	AWT	Answered rate	QoS
1 week										
Tuesday	0	0	0	0	0	0	0			
Wednesday	10	0	0	0	0	0	10	00:00:00	0 %	
Thursday	13	12	1	0	0	0	0	00:00:23	92 %	0 %
Friday	19	17	2	0	0	0	0	00:00:25	89 %	0 %
2 week										
Monday	28	27	0	0	0	0	1	00:00:10	96 %	0 %
Tuesday	22	20	1	0	0	0	1	00:00:04	90 %	0 %
Wednesday	23	19	2	0	0	0	2	00:01:25	82 %	0 %
Thursday	21	21	0	0	0	0	0	00:00:07	100 %	0 %
Friday	34	30	1	0	0	0	3	00:00:08	88 %	0 %
3 week										
Monday	36	35	0	0	0	0	1	00:00:11	97 %	0 %
Tuesday	40	36	4	0	0	0	0	00:00:07	90 %	0 %
Wednesday	35	35	0	0	0	0	0	00:00:07	100 %	0 %
Thursday	51	51	0	0	0	0	0	00:00:05	100 %	0 %
Friday	16	16	0	0	0	0	0	00:00:04	100 %	0 %
4 week										
Monday	24	24	0	0	0	0	0	00:00:04	100 %	0 %
Tuesday	25	24	1	0	0	0	0	00:00:08	96 %	0 %
Wednesday	28	24	4	0	0	0	0	00:00:26	85 %	0 %
Thursday	40	37	0	3	0	0	0	00:00:09	100 %	0 %
Friday	39	37	2	0	0	0	0	00:00:06	94 %	0 %
5 week										
Monday	43	43	0	0	0	0	0	00:00:07	100 %	0 %
Tuesday	35	32	1	0	0	0	2	00:00:06	91 %	0 %
Wednesday	31	28	1	0	0	0	2	00:00:07	90 %	0 %
Thursday	27	15	1	0	0	0	11	00:00:49	55 %	0 %
Total	640	583	21	3	0	0	33	00:00:13	91 %	0 %

				Dissuaded or Overflowed						
	Received	Answered	Abandoned	Closed	Saturated	NA	Blocking	AWT	Answered rate	QoS
January	0	0	0	0	0	0	0			
February	0	0	0	0	0	0	0			
March	0	0	0	0	0	0	0			
April	0	0	0	0	0	0	0			
May	0	0	0	0	0	0	0			
June	0	0	0	0	0	0	0			
July	119	98	7	13	0	0	1	00:00:08	92 %	1 %
August	95	72	6	3	0	0	14	00:00:10	78 %	0 %
September	181	153	9	0	0	0	19	00:00:11	84 %	0 %
October	214	159	9	0	0	0	46	00:00:13	74 %	0 %
November	177	169	6	1	0	0	1	00:00:06	96 %	0 %
December	194	180	5	1	0	0	8	00:00:07	93 %	0 %
Total	973	824	42	18	0	0	89	00:00:09	86 %	0 %

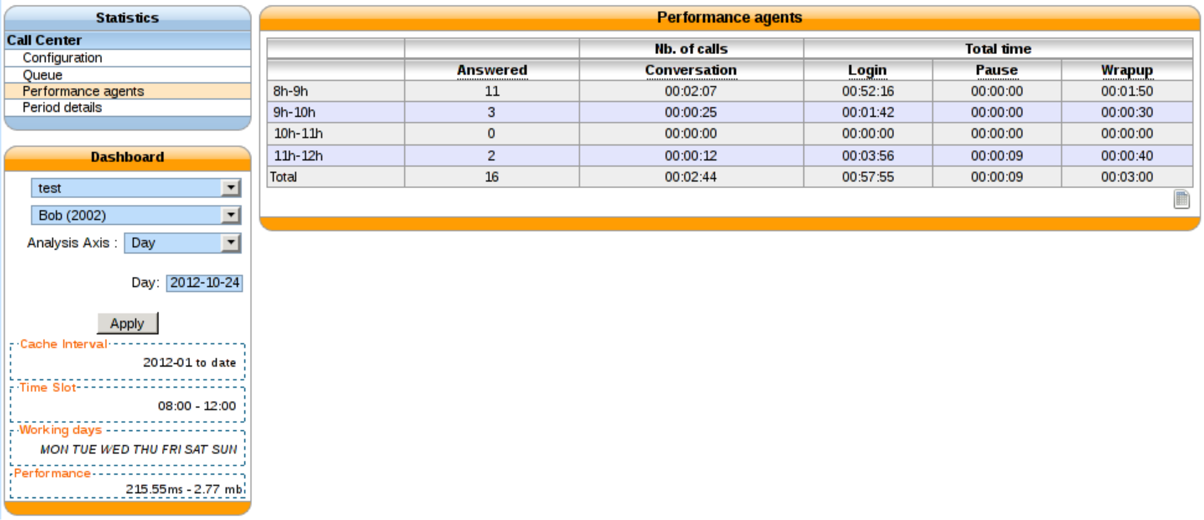


   - Blocking : Calls received when no agents were available or when there were no agents to take the call, join an empty queue condition or remove callers if there are no agents condition is reached (advanced queue parameter tab).

- Average waiting time (AWT): The average waiting time of call on wait
- Answered rate (HR): The ratio of answered calls over (received calls - closed calls)
- Quality of service (QoS): Percentage of calls answered in less than x seconds over the number of answered calls, where x is defined in the configuration

Agent performance

Agent performance statistics can be viewed in *Services* → *Statistics* → *Performance agents*.



Note: The agent performance counters do not take into account transfer between agents: if agent A processes a call and transfers it to agent B, only the counters of agent A will be updated. Ignoring any info after the call transfer.

Counters

- Answered: Number of calls answered by the agent
- Conversation: Total time spent for calls answered during a given period
- Login: Total login time of an agent.
- Wrapup: Total time spent in wrapup by an agent.
- Pause: Total pause time of an agent

Warning: Data generated before XiVO 12.19 might have erroneous results for the Login time counter

Note: The Pause time counter only supports **PAUSEALL** and **UNPAUSEALL** command from ctclient

Note: Wrapup time events were added to XiVO in version 12.21

Performance agents

	Nb. of calls	Total time			
	Answered	Conversation	Login	Pause	Wrapup
Karine Boudoix (108)	87	09:59:57	30:58:51	10:53:16	00:21:30
Fred Epric (100)	45	01:27:52	34:59:45	05:32:45	00:07:30
Hipolyte Marroussou (102)	13	00:24:23	27:42:47	97:50:42	00:02:10
Gérard Mensour (101)	0	00:00:00	00:00:00	00:00:00	00:00:00
Irène Pourtox (106)	39	03:52:40	34:08:47	22:39:31	00:09:45
Juliette Queriau (107)	78	09:03:51	34:06:24	91:23:52	00:19:30

Agent summary

Agent per day

	Nb. of calls	Total time			
	Answered	Conversation	Login	Pause	Wrapup
9h-10h	1	00:07:31	00:28:45	00:55:13	00:00:00
10h-11h	5	00:30:50	01:00:00	00:00:00	00:01:15
11h-12h	4	00:24:15	01:00:00	00:00:20	00:01:15
12h-13h	0	00:00:00	00:33:34	00:59:07	00:00:00
13h-14h	0	00:00:00	01:00:00	00:34:20	00:00:00
14h-15h	3	01:23:38	01:00:00	00:00:00	00:00:30
15h-16h	2	00:05:24	01:00:00	00:00:00	00:00:30
16h-17h	3	00:26:21	01:00:00	00:00:00	00:01:00
17h-18h	6	00:37:50	01:00:00	00:00:00	00:01:15
Total	24	03:35:49	08:02:20	02:29:01	00:05:45

Agent per week

	Nb. of calls	Total time			
	Answered	Conversation	Login	Pause	Wrapup
Monday 1	24	03:35:49	08:02:20	02:29:01	00:05:45
Tuesday 2	22	02:17:11	07:31:53	03:36:46	00:05:30
Wednesday 3	19	01:40:33	07:27:13	02:34:03	00:04:45
Thursday 4	22	02:26:24	07:57:25	02:13:23	00:05:30
Friday 5	0	00:00:00	00:00:00	00:00:00	00:00:00
Total	87	09:59:57	30:58:51	10:53:16	00:21:30

Agent per month

Agent per year

Period details

Display by period defined in configuration, i.e. between 0 and 10s, 10s and 30s etc ... the number of handled calls and the number of abandonned calls

You may click on a queue name to get more information for this queue

	Nb. of calls	Total time			
	Answered	Conversation	Login	Pause	Wrapup
23 week					
Monday	0	00:00:00	09:00:00	00:00:00	00:00:00
Tuesday	0	00:00:00	09:00:00	00:00:00	00:00:00
Wednesday	0	00:00:00	09:00:00	00:00:00	00:00:00
Thursday	0	00:00:00	09:00:00	00:00:00	00:00:00
Friday	0	00:00:00	09:00:00	00:00:00	00:00:00
24 week					
Monday	0	00:00:00	09:00:00	00:00:00	00:00:00
Tuesday	0	00:00:00	03:54:46	00:00:00	00:00:00
Wednesday	0	00:00:00	00:00:00	00:00:00	00:00:00
Thursday	0	00:00:00	04:35:05	04:56:29	00:00:00
Friday	10	01:23:58	08:24:59	03:11:31	00:01:35
25 week					
Monday	15	01:20:19	08:17:50	03:02:58	00:02:30
Tuesday	3	00:13:52	08:22:51	07:06:44	00:00:20
Wednesday	3	00:20:02	08:28:03	06:12:39	00:00:40
Thursday	0	00:00:00	08:24:06	08:24:01	00:00:00
Friday	1	00:09:22	08:27:27	05:27:33	00:00:15
26 week					
Monday	10	00:36:41	08:23:33	04:48:20	00:02:30
Tuesday	11	01:08:46	08:30:00	04:34:26	00:02:45
Wednesday	3	00:07:48	08:29:34	04:58:42	00:00:45
Thursday	5	00:40:10	04:00:58	07:26:52	00:01:15
Friday	9	01:12:33	08:19:18	04:27:04	00:02:15
Total	70	07:13:31	150:38:3	64:37:24	00:14:50

	Nb. of calls	Total time			
	Answered	Conversation	Login	Pause	Wrapup
January	0	00:00:00	00:00:00	00:00:00	00:00:00
February	0	00:00:00	00:00:00	00:00:00	00:00:00
March	0	00:00:00	00:00:00	00:00:00	00:00:00
April	0	00:00:00	00:00:00	00:00:00	00:00:00
May	1	00:00:34	97:17:07	00:00:00	00:00:05
June	89	09:01:48	159:03:1	69:17:25	00:19:35
July	39	03:52:40	34:08:47	22:39:31	00:09:45
August	0	00:00:00	00:00:00	00:00:00	00:00:00
September	0	00:00:00	00:00:00	00:00:00	00:00:00
October	0	00:00:00	00:00:00	00:00:00	00:00:00
November	0	00:00:00	00:00:00	00:00:00	00:00:00
December	0	00:00:00	00:00:00	00:00:00	00:00:00
Total	110	11:06:45	282:04:2	87:16:55	00:24:40

Details by time

P1: 0-15s | P2: 16-20s | P3: 21-30s | P4: 31-60s | P5: 61s |

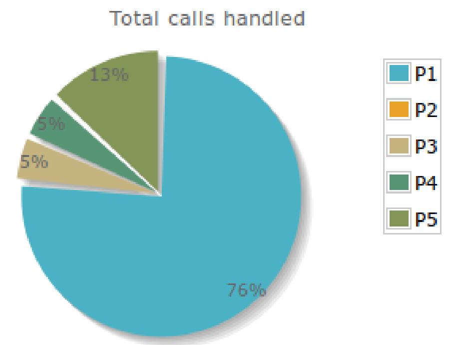
	Calls handled					Abandoned calls				
	P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
blue	1992	25	140	152	250	38	3	6	15	26
green	0	0	0	0	0	0	0	0	0	0
red	0	0	0	0	0	0	0	0	0	0
yellow	640	2	18	7	2	10	1	0	2	1

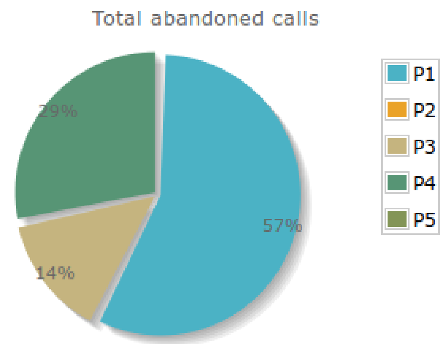
Period details by day

Details by time

P1: 0-15s | P2: 16-20s | P3: 21-30s | P4: 31-60s | P5: 61s |

	Calls handled					Abandoned calls				
	P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
8h-9h	0	0	0	0	0	0	0	0	0	0
9h-10h	13	0	1	5	12	0	0	0	0	0
10h-11h	16	0	2	0	5	2	0	0	2	0
11h-12h	20	0	0	0	1	1	0	0	0	0
12h-13h	0	0	0	0	0	0	0	0	0	0
13h-14h	10	0	1	0	3	0	0	1	0	0
14h-15h	26	0	0	1	0	0	0	0	0	0
15h-16h	26	0	5	2	2	1	0	0	0	0
16h-17h	18	0	1	0	1	0	0	0	0	0
17h-18h	18	0	0	2	2	0	0	0	0	0
18h-19h	0	0	0	0	0	0	0	0	0	0
19h-20h	0	0	0	0	0	0	0	0	0	0
Total	147	0	10	10	26	4	0	1	2	0





Period details by week

Period details by month

Period details by year

1.9.7 Reporting

You may use your own reporting tools to be able to produce your own reports provided **you do not use the XiVO server original tables**, but copy the tables to your own data server. You may use the following procedure as a template :

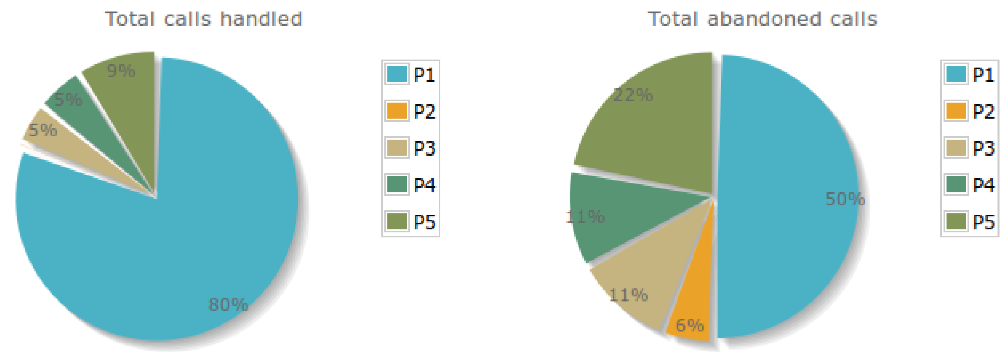
- Allow remote database access on XiVO
- Create a postgresql account read only on asterisk database
- Create target tables in your database located on the data server
- Copy the statistic table content to your data server

General Architecture

1. The *queue_log* table of the *asterisk* database is filled by events from Asterisk and by custom dialplan events

P1: 0-15s | P2: 16-20s | P3: 21-30s | P4: 31-60s | P5: 61s |

	Calls handled					Abandoned calls				
	P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
Monday 1	147	0	10	10	26	4	0	1	2	0
Tuesday 2	145	2	8	8	6	2	0	0	0	1
Wednesday 3	128	0	8	7	7	1	0	1	0	2
Thursday 4	122	2	5	11	23	2	1	0	0	1
Friday 5	0	0	0	0	0	0	0	0	0	0
Total	542	4	31	36	62	9	1	2	2	4



2. *xivo-stat fill_db* is then used to read data from the *queue_log* table and generate the tables *stat_call_on_queue* and *stat_queue_periodic*
3. The web interface generate tables and graphics from the *stat_call_on_queue* and *stat_queue_periodic* tables depending on the selected configuration

Statistic Data Table Content

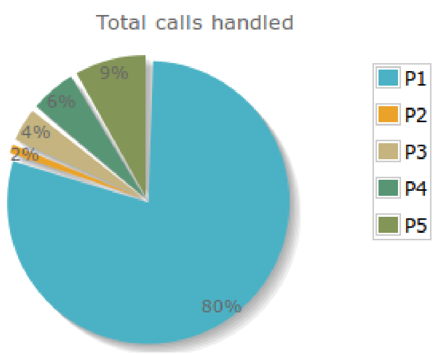
stat_call_on_queue

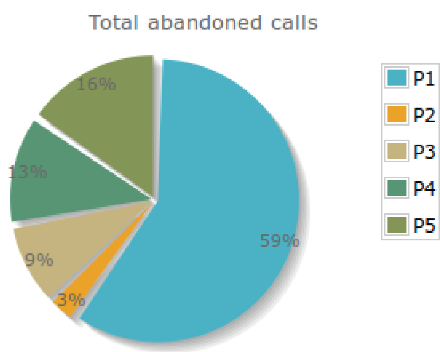
This table is used to store each call individually. Each call received on a queue generates a single entry in this table containing time related fields and a foreign key to the agent who answered the call and another on the queue on which the call was received.

It also contains the status of the call ie. answered, abandoned, full, etc.

P1: 0-15s | P2: 16-20s | P3: 21-30s | P4: 31-60s | P5: 61s |

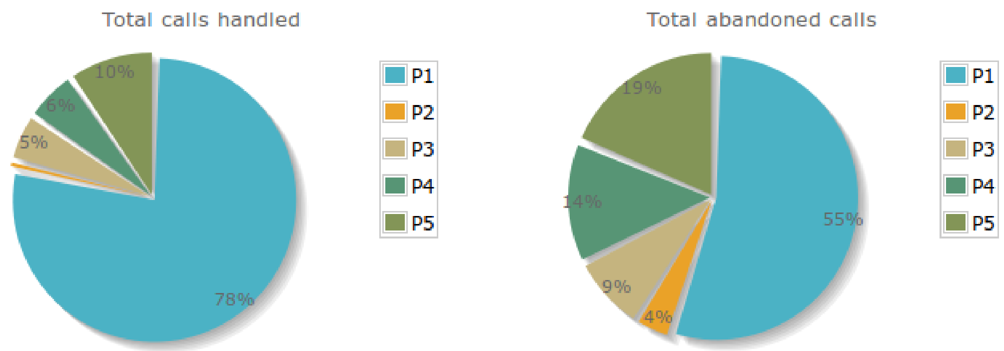
	Calls handled					Abandoned calls				
	P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
27 week										
Monday	147	0	10	10	26	4	0	1	2	0
Tuesday	145	2	8	8	6	2	0	0	0	1
Wednesday	128	0	8	7	7	1	0	1	0	2
Thursday	122	2	5	11	23	2	1	0	0	1
Friday	0	0	0	0	0	0	0	0	0	0
28 week										
Monday	0	0	0	0	0	0	0	0	0	0
Tuesday	0	0	0	0	0	0	0	0	0	0
Wednesday	0	0	0	0	0	0	0	0	0	0
Thursday	0	0	0	0	0	0	0	0	0	0
Friday	0	0	0	0	0	0	0	0	0	0
29 week										
Monday	0	0	0	0	0	0	0	0	0	0
Tuesday	0	0	0	0	0	0	0	0	0	0
Wednesday	27	7	0	5	0	10	0	1	2	1
Thursday	0	0	0	0	0	0	0	0	0	0
Friday	0	0	0	0	0	0	0	0	0	0
30 week										
Monday	0	0	0	0	0	0	0	0	0	0
Tuesday	0	0	0	0	0	0	0	0	0	0
Wednesday	0	0	0	0	0	0	0	0	0	0
Thursday	0	0	0	0	0	0	0	0	0	0
Friday	0	0	0	0	0	0	0	0	0	0
31 week										
Monday	0	0	0	0	0	0	0	0	0	0
Tuesday	0	0	0	0	0	0	0	0	0	0
Wednesday	0	0	0	0	0	0	0	0	0	0
Total	569	11	31	41	62	19	1	3	4	5





P1: 0-15s | P2: 16-20s | P3: 21-30s | P4: 31-60s | P5: 61s |

	Calls handled					Abandoned calls				
	P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
January	0	0	0	0	0	0	0	0	0	0
February	21	0	0	1	7	28	4	9	9	5
March	10	0	0	0	0	10	0	0	0	0
April	4	0	0	0	0	16	0	0	0	0
May	1	0	0	0	0	3	0	0	0	2
June	1570	14	119	121	214	23	2	4	13	21
July	569	11	31	41	62	19	1	3	4	5
August	0	0	0	0	0	0	0	0	0	0
September	0	0	0	0	0	0	0	0	0	0
October	0	0	0	0	0	0	0	0	0	0
November	0	0	0	0	0	0	0	0	0	0
December	0	0	0	0	0	0	0	0	0	0
Total	2028	25	140	153	257	95	7	15	24	33



Field	Values	Description
id	generated	
callid	numeric value	This call id is also used in the CEL table and can be used to get call detail information
time	Call time	
ringtime		Ringing duration time in seconds
talktime		Talk time duration in seconds
waittime		Wait time duration in seconds
status		See status description below
queue_id		Id of the queue, the name of the queue can be found in table `stat_queue`, using this name queue details can be found in table `queuefeatures`
agent_id		Id of the agent, the agent name can be found in table `stat_agent`, using this name agent details can be found in table `agentfeatures` using the number in the second part of the name Exemple : Agent/1002 is agent with number 1002 in table `agentfeatures`

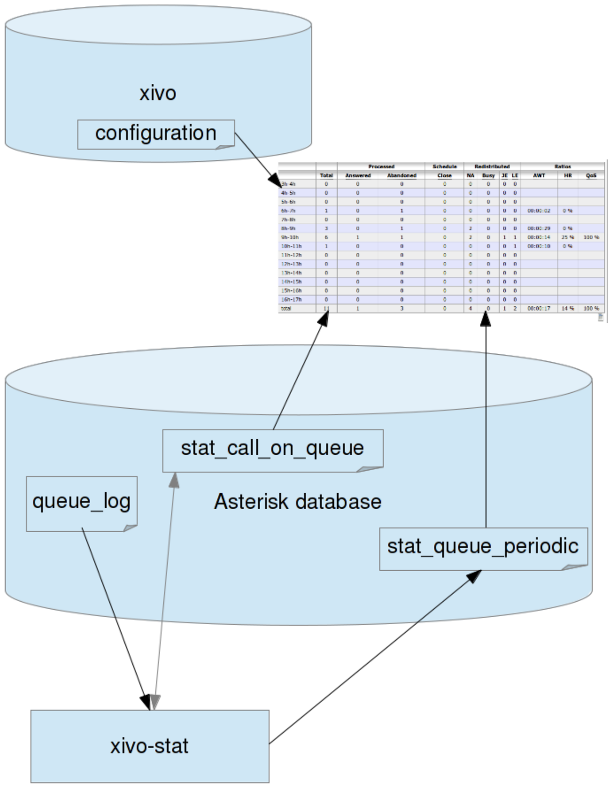


Figure 1.84: Statistics Architecture

Queue Call Status

Status	Description
full	Call was not queued because queue was full, happens when the number of calls is greater than the maximum number of calls allowed to wait
closed	Closed due to the schedule applied to the queue
joinempty	No agents were available in the queue to take the call (follows the join empty parameter of the queue)
leaveempty	No agents available while the call was waiting in the qeueue
divert_ca_ratio	Call diverted because the ratio number of agent number of calls waiting configured was exceeded
divert_waittime	Call diverted because the maximum expected waiting time configured was exceeded
answered	Call was answered
abandoned	Call hangup by the caller
timeout	Call stayed longer than the maximum time allowed in queue parameter

stat_queue_periodic Table

This table is an aggregation of the queue_log table.

This table contains counters on each queue for each given period. The granularity at the time of this writing is an hour and is not configurable. This table is then used to compute statistics for a given range of hours, days, week, month or year.

Field	Description
id	Generated id
time	time period, all counters are aggregated for an hour
answered	Number of answered calls during the period
abandoned	Number of abandoned calls during the period
total	Total calls received during the period
full	Number of calls received when queue was full
closed	Number of calls received on close
joinempty	Number of calls received no agents available
leaveempty	Number of calls diverted agents not available during the wait
divert_ca_ratio	Number of calls diverted due to the number of agent number versus calls waiting configured was exceeded
divert_waittime	Number of calls diverted because the maximum expected waiting time configured was exceeded
timeout	Number of calls diverted because the maximum time allowed in queue parameter was exceeded
queue_id	

stat_agent

This table is used to match agents to an id that is different from the id in the agent configuration table. This is necessary to avoid loosing statistics on a deleted agent. This also means that if an agent changes number ie. Agent/1001 to Agent/1202, the supervisor will have to take this information into account when viewing the statistics. Affecting an old number to a another agent also means that the supervisor will have to ignore entries for this given agent for the period before the number assignment to the new agent.

stat_queue

This table is used to store queues in a table that is different from the queue configuration table. This is necessary to avoid losing statistics on a deleted queue. Renaming a queue is also not handled at this time.

1.10 High Availability (HA)

The HA (High Availability) solution in XiVO makes it possible to maintain basic telephony function whatever your main XiVO server is running or not. When running a XiVO HA cluster, users are guaranteed to never experience a downtime of more than 5 minutes of their basic telephony service.

The HA solution in XiVO is based on a 2-nodes "master and slave" architecture. In the normal situation, both the master and slave nodes are running in parallel, the slave acting as an "hot standby", and all the telephony services are provided by the master node. If the master fails or must be shutdown for maintenance, then the telephony devices automatically communicate with the slave node instead of the master one. Once the master is up again, the telephony devices failback to the master node. Both the failover and the failback operation are done automatically, i.e. without any user intervention, although an administrator might want to run some manual operations after failback as to, for example, make sure any voicemail messages that were left on the slave are copied back to the master.

1.10.1 Prerequisites

The HA in XiVO only works with telephony devices (i.e. phones) that support the notion of a primary and backup telephony server.

- The master and the slave must be in the same subnet
- If firewalling, the master must be allowed to join the slave on port 5432
- Trunk registration timeout (`expiry`) should be less than 300 seconds (5 minutes)

The HA solution is guaranteed to work correctly with *the following devices*.

1.10.2 Quick Summary

- You need two configured XiVO (wizard passed)
- Configure one XiVO as a master -> setup the slave address
- Restart services (xivo-service restart) on master
- Configure the other XiVO as a slave -> setup the master address
- Start configuration synchronization by running the script `xivo-master-slave-db-replication <slave_ip>` on the master
- Resynchronize all your devices
- Configure the XiVO Clients

That's it you now have a HA configuration, and every hour all the configuration done on the master will be reported to the slave.

1.10.3 Configuration Details

First thing to do is to *install 2 XiVO*.

Important: When you upgrade a node of your cluster, you must also upgrade the other so that they both are running the same version of XiVO. Otherwise, the replication might not work properly.

You must configure the HA in the Web interface (*Configuration* → *Management* → *High Availability* page).

You can configure the master and slave in whatever order you want.

Warning: When the HA is configured, some changes will be automatically made to the configuration of XiVO.

SIP expiry value on master and slave will be automatically updated:

- min: 3 minutes
- max: 5 minutes
- default: 4 minutes

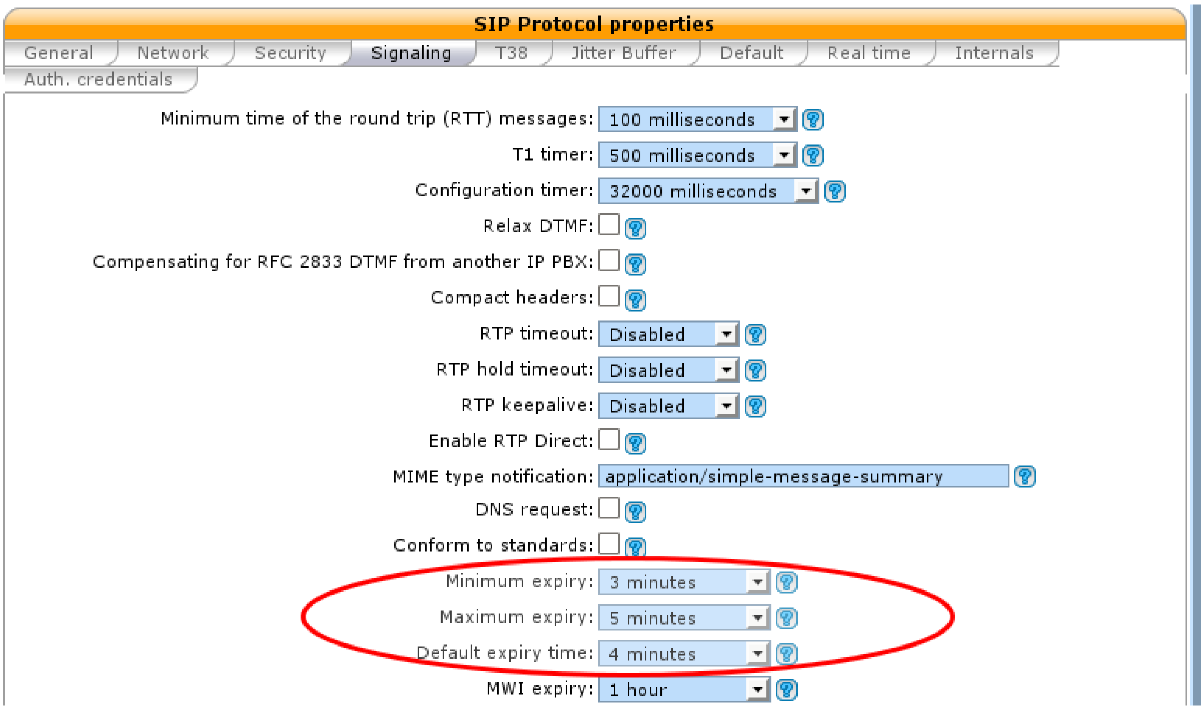


Figure 1.85: *Services → IPBX → General Settings → SIP Protocol*

The provisioning server configuration will be automatically updated in order to allow phones to switch from XiVO power failure.

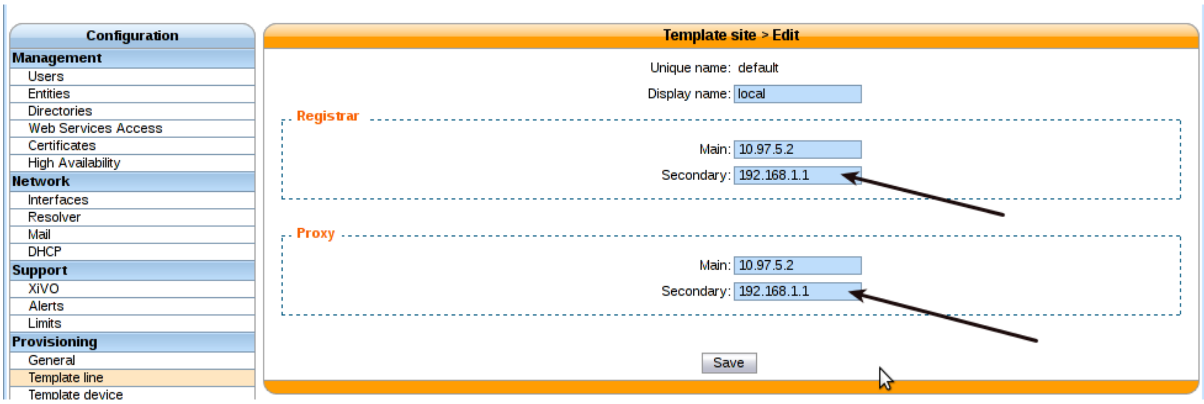


Figure 1.86: *Configuration → Provisioning → Template Line → Edit default*

Warning: Especially not change these values when the HA is configured, this could cause problems. These values will be reset to blank when the HA is disabled.

Important: For the telephony devices to take the new proxy/registrar settings into account, you must *resynchronize the devices* or restart them manually.

Disable node

Default status of HIGH AVAILABILITY (HA) is disabled:

Note: You can reset at any time by choosing a server mode (disabled)

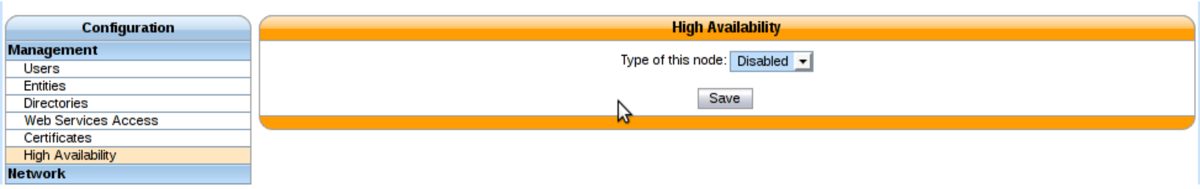


Figure 1.87: HA Dashboard Disabled (default state)

Important: You have to restart services (xivo-service restart) once the master node is disabled.

Master node

In choosing the method `Master` you must enter the IP address of the slave node.

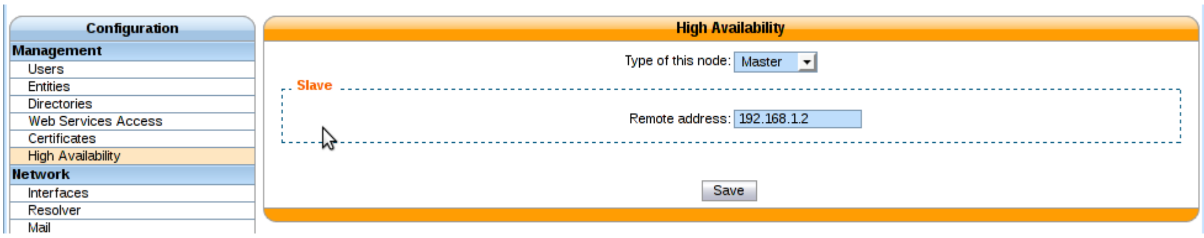


Figure 1.88: HA Dashboard Master

Important: You have to restart all services (xivo-service restart) once the master node is configured.

Slave node

In choosing the method `Slave` you must enter the IP address of master node.

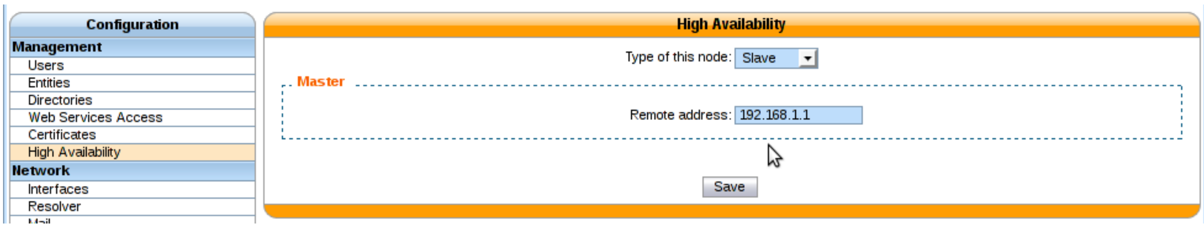


Figure 1.89: HA Dashboard Slave

Configuration Replication

Once master slave configuration is completed, XiVO configuration is replicated from the master node to the slave every hour (:00). Replication can be started manually by running the replication script on the master:

```
xivo-master-slave-db-replication <slave_ip>
```

The replication does not copy the full XiVO configuration of the master. Notably, these are excluded:

- Call logs
- Call center statistics
- All the network configuration (i.e. everything under the *Configuration* → *Network* section)
- All the support configuration (i.e. everything under the *Configuration* → *Support* section)
- HA settings

Less importantly, these are also excluded:

- Queue logs
- CELs

The replication only includes a (partial) replication of the database used by XiVO, so everything that is stored outside the database is also not copied. Here's an non exhaustive list of things that are not stored in the database, and thus are not copied:

- Certficates
- Audio files
- On-hold music
- Custom dialplan
- Voicemail messages
- Provisioning configuration

XiVO Client

You have to enter the master and slave address in the `Connection` tab of the XiVO Client configuration :

The main server is the master node and the backup server is the slave node.

When connecting the XiVO Client with the main server down, the login screen will hang for 3 seconds before connecting to the backup server.

1.10.4 Internals

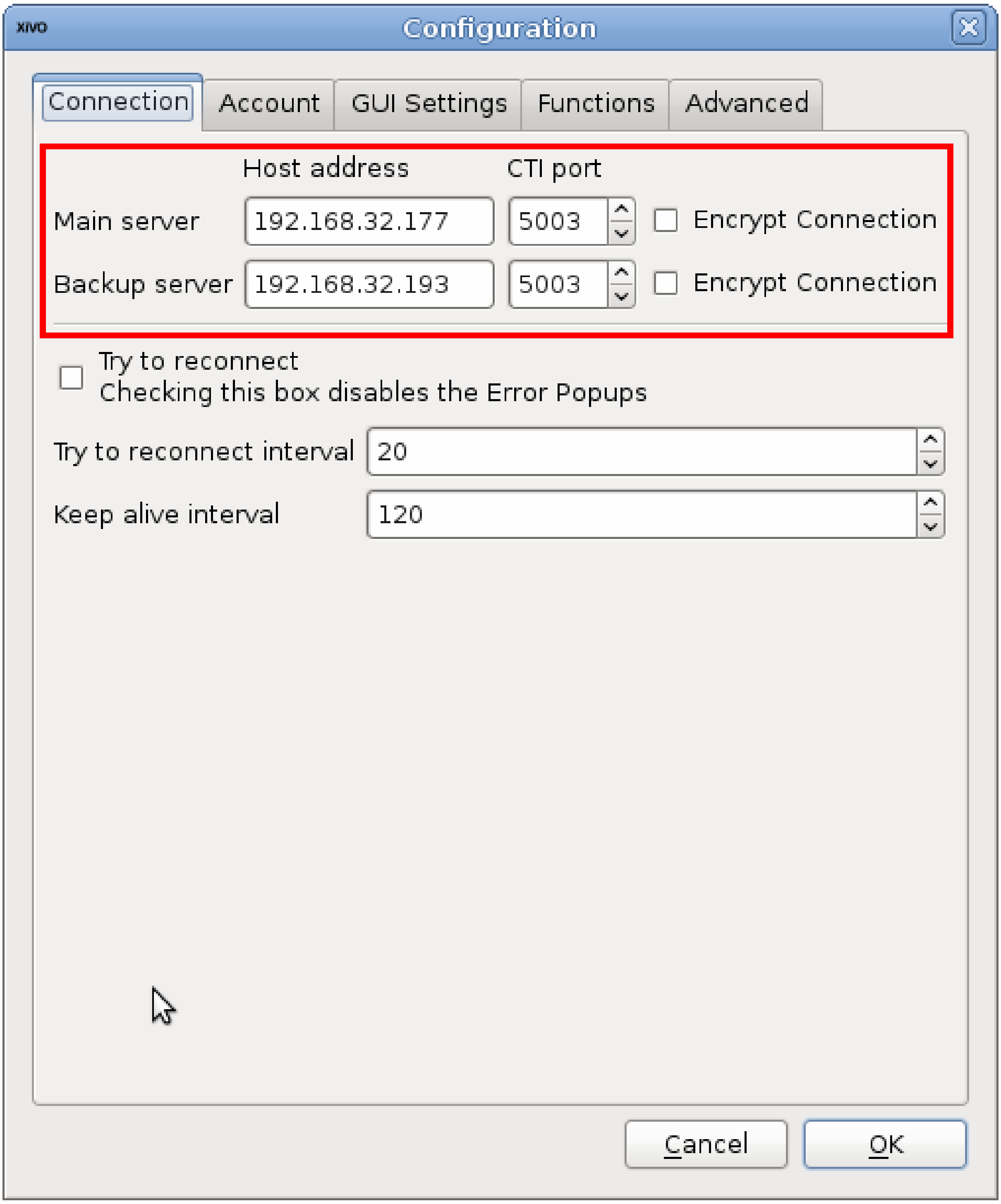
3 scripts are used to manage services and data replication.

- xivo-master-slave-db-replication <slave_ip> is used on the master to replicate the master's data on the slave server. It runs on the master.
- xivo-manage-slave-services {start,stop} is used on the slave to start, stop monit and asterisk. The services won't be restarted after an upgrade or restart.
- xivo-check-master-status <master_ip> is used to check the status of the master and enable or disable services accordingly.

1.10.5 Limitations

When the master node is down, some features are not available and some behave a bit differently. This includes:

- Call history / call records are not recorded.
- Voicemail messages saved on the master node are not available.
- Custom voicemail greetings recorded on the master node are not available.


Configuration
Connection
Account
GUI Settings
Functions
Advanced
Host address
CTI port
Main server
192.168.32.177
5003
Encrypt Connection
Backup server
192.168.32.193
5003
Encrypt Connection
Try to reconnect
Checking this box disables the Error Popups
Try to reconnect interval
20
Keep alive interval
120
Cancel
OK

- More generally, custom sounds are not available. This includes music on hold and recordings.
- Custom dialplan (i.e. dialplan found in the `/etc/asterisk/extensions_extra.d` directory or in the *Services → IPBX → IPBX configuration → Configuration files* page) is not available.

Note that, on failover and on failback:

- DND, call forwards, call filtering, ..., statuses may be lost if changed recently.
- If you are connected as an agent, then you might need to reconnect as an agent when the master goes down. Since it's hard to know when the master goes down, if your CTI client disconnect and you can't reconnect it, then it's a sign the master might be down.

Additionally, only on failback:

- Voicemail messages are not copied from the slave to the master, i.e. if someone left a message on your voicemail when the master was down, you won't be able to consult it once the master is up again.
- More generally, custom sounds are not copied back. This includes recordings.

Here's the list of limitations that are more relevant on an administrator standpoint:

- In the case a DHCP server is running on the master node, then when the master is down, phones won't be able to get a new DHCP lease, so it is advised not to restart the phones.
- The master status is up or down, there's no middle status. This mean that if Asterisk is crashed the XiVO is still up and the failover will NOT happen.

1.10.6 Berofos Integration

Berofos Integration

XiVO offers the possibility to integrate a berofos failover switch within a HA cluster.

This is useful if you have one or more ISDN lines (i.e. T1/E1 or T0 lines) that you want to use whatever the state of your XiVO HA cluster. To use a berofos within your XiVO HA installation, you need to properly configure both your berofos and your XiVOs, then the berofos will automatically switch your ISDN lines from your master node to your slave node if your master goes down, and vice-versa when it comes back up.

You can also use a Berofos failover switch to secure the ISDN provider lines when installing a XiVO in front of an existing PBX. The goal of this configuration is to mitigate the consequences of an outage of the XiVO : with this equipment the ISDN provider links could be switched to the PBX directly if the XiVO goes down.

XiVO **does not offer natively** the possibility to configure Berofos in this failover mode. The *Berofos Integration with PBX* section describes a workaround.

Installation and Configuration

Master Configuration There is nothing to be done on the master node.

Slave Configuration First, install the bntools package:

```
apt-get install bntools
```

This will make the `bnfos` command available.

You can then connect your berofos to your network and power it on. By default, the berofos will try to get an IP address via DHCP. If it is not able to get such address from a DHCP server, it will take the 192.168.0.2/24 IP address.

Note: The DHCP server on XiVO does not offer IP addresses to berofos devices by default.

Next step is to create the `/etc/bnfos.conf` file via the following command:

```
bnfos --scan -x
```

If no berofos device is detected using this last command, you'll have to explicitly specify the IP address of the berofos via the -h option:

```
bnfos --scan -x -h <berofos ip>
```

At this stage, your `/etc/bnfos.conf` file should contains something like this:

```
[fos1]
mac = 00:19:32:00:12:1D
host = 10.34.1.50
#login = <user>:<password>
```

It is advised to configure your berofos with a static IP address. You first need to put your berofos into *flash mode* :

- press and hold the black button next to the power button,
- power on your berofos,
- release the black button when the red LEDs of port D start blinking.

Then, you can issue the following command, by first replacing the network configuration with your one:

```
bnfos --netconf -f fos1 -i 10.34.1.20 -n 255.255.255.0 -g 10.34.1.1 -d 0
```

Note:

- `-i` is the IP address
- `-n` is the netmask
- `-g` is the gateway
- `-d 0` is to disable DHCP

You can then update your berofos firmware to version 1.53:

```
wget http://www.beronet.com/downloads/berofos/bnfos_v153.bin
bnfos --flash bnfos_v153.bin -f fos1
```

Once this is done, you'll have to reboot your berofos in operationnal mode (that is in normal mode).

Then you must rewrite the `/etc/bnfos.conf` (mainly if you changed the IP address):

```
bnfos --scan -x -h <berofos ip>
```

Now that your berofos has proper network configuration and an up to date firmware, you might want to set a password on your berofos:

```
bnfos --set apwd=<password> -f fos1
bnfos --set pwd=1 -f fos1
```

You must then edit the `/etc/bnfos.conf` and replace the login line to something like:

```
login = admin:<password>
```

Next, configure your berofos for it to work correctly with the XiVO HA:

```
bnfos --set wdog=0 -f fos1
bnfos --set wdogdef=0 -f fos1
bnfos --set scenario=0 -f fos1
bnfos --set mode=1 -f fos1
bnfos --set modedef=1 -f fos1
```

This, among other things, disable the watchdog. The switching from one relay mode to the other will be done by the XiVO slave node once it detects the master node is down, and vice-versa.

Finally, you can make sure everything works fine by running the xivo-berofos command:

```
xivo-berofos master
```

The green LEDs on your berofos should be lighted on ports A and B.

Connection

Two XiVOs Here's how to connect the ISDN lines between your berofos with:

- two XiVOs in high availability

In this configuration you can protect **up two 4** ISDN lines. If more than 4 ISDN lines to protect, you must set up a Multiple berofos configuration.

Here's an example with 4 ISDN lines coming from your telephony provider:

```
ISDN lines (provider)
  | | | |
  | | | |
+---------------------------------------------+
|    A           B           C           D    |
| 1|2|3|4     1|2|3|4     1|2|3|4     1|2|3|4 |
+---------------------------------------------+
              | | | |                 | | | |
              | | | |                 | | | |
            +--------+              +--------+
            | xivo-1 |              | xivo-2 |
            +--------+              +--------+
```

Two XiVOs and one PBX Here's how to connect your berofos with:

- two XiVOs in high availability,
- one PBX.

In this configuration you can protect **up two 2** ISDN lines. If more than 2 ISDN lines to protect, you must set up a Multiple berofos configuration.

Logical view:

```
                +--------+                            +-----+
-- Provider ----| xivo-1 | -- ISDN Interconnection  --| PBX | -- Phones
                +--------+                            +-----+
                  | xivo-2 |
                  +--------+
```

This example shows the case where there are 2 ISDN lines coming from your telephony provider:

```
ISDN lines (provider)
  | |
  | |
+------------------------------------------------------+
|    A               B            C           D        |
| 1|2|3|4         1|2   3|4      1|2|3|4     1|2   3|4 |
+------------------------------------------------------+
      | |     CPE | |   | | NET          CPE | |   | | NET
      | |   spans | |   | | spans      spans | |   | | spans
      | |       +----------+              +------------+
      | |       |  xivo-1  |              |   xivo-2   |
      | |       +----------+              +------------+
      | |
      | |
    +------+
```

```
| PBX   |
+-------+
```

One XiVO and one PBX This case is not currently supported. You'll find a workaround in the *Berofos Integration with PBX* section.

Multiple berofos It's possible to use more than 1 berofos with XiVO.

For each supplementary berofos you want to use, you must first configure it properly like you did for the first one. The only difference is that you need to add a berofos declaration to the `/etc/bnfos.conf` file instead of creating/overwriting the file. Here's an example of a valid config file for 2 berofos:

```
[fos1]
mac = 00:19:32:00:12:1D
host = 10.100.0.201
login = admin:foobar

[fos2]
mac = 00:11:22:33:44:55
host = 10.100.0.202
login = admin:barfoo
```

Warning: berofos name must follow the pattern `fosX` where X is a number starting with 1, then 2, etc. The `bnfos` tool won't work properly if it's not the case.

Operation

When your XiVO switch the relay mode of your berofos, it logs the event in the `/var/log/syslog` file.

Default mode

Note that when the berofos is off, the A and D ports are connected together. This behavior is not customizable.

Uninstallation

It is important to remove the `/etc/bnfos.conf` file on the slave node when you don't want to use anymore your berofos with your XiVOs.

Reset the Berofos

You can reset the berofos configuration :

1. Power on the berofos,
2. When red and green LEDs are still lit, press & hold the black button,
3. Release it when the red LEDs of the D port start blinking fast
4. Reboot the beronet, it should have lost its configuration.

External links

- Install BeroFos (in French)
- berofos user manual

1.11 API and SDK

1.11.1 Channel Event Logging (CEL)

CELs are the mechanism used by Asterisk to store telephony events. XiVO uses CELs instead of CDRs since they are more accurate and flexible.

Official Documentation

Asterisk's CEL documentation can be accessed online on the Asterisk wiki.

CEL Samples

The following configuration will be used for the following examples.

- User *Tux* with extension *1000*
- User *Père Noël* with extension *1001*
- External User with number *555-555-5555*
- External calls to DID *445* are sent to *1000*
- External calls to DID *446* are sent to *1001*
- Our DID number is *555-555-6666*

Outgoing Call

1. 1000 *calls* 555-555-5555
2. 555-555-5555 *answers*
3. 555-555-5555 *hangs-up* after a few seconds

```
  id   |   eventtype    |          eventtime          | userdeftype |    cid_name     |  cid_num     | c
-------+----------------+-----------------------------+-------------+-----------------+-------------+--
 54048 | CHAN_START     | 2012-03-29 09:50:00.295885 |             | Tux             | 1000         |
 54049 | APP_START      | 2012-03-29 09:50:00.322427 |             | 5555556666      | 5555556666 | 1
 54050 | CHAN_START     | 2012-03-29 09:50:00.322539 |             |                 |              |
 54051 | ANSWER         | 2012-03-29 09:50:06.680481 |             |                 | dial         |
 54052 | ANSWER         | 2012-03-29 09:50:06.680634 |             | 5555556666      | 5555556666 | 1
 54053 | BRIDGE_START   | 2012-03-29 09:50:06.680678 |             | 5555556666      | 5555556666 | 1
 54054 | BRIDGE_END     | 2012-03-29 09:50:13.90645  |             | 5555556666      | 5555556666 | 1
 54055 | HANGUP         | 2012-03-29 09:50:13.906764 |             |                 | dial         |
 54056 | CHAN_END       | 2012-03-29 09:50:13.906803 |             |                 | dial         |
 54057 | HANGUP         | 2012-03-29 09:50:13.906855 |             | 5555556666      | 5555556666 | 1
 54058 | CHAN_END       | 2012-03-29 09:50:13.907034 |             | 5555556666      | 5555556666 | 1
(11 rows)
```

Internal Call

1. 1000 calls 1001
2. 1001 *answers*
3. 1001 *hangs up* after a few seconds

```
  id  |   eventtype    |          eventtime           | userdeftype | cid_name   | cid_num | cid_ani  |
------+----------------+-----------------------------+-------------+------------+---------+----------+
 8527 | CHAN_START     | 2012-03-28 13:43:55.379674 |             | Tux        | 1000    |          |
 8528 | APP_START      | 2012-03-28 13:43:55.435472 |             | Tux        | 1000    | 1000     |
 8529 | CHAN_START     | 2012-03-28 13:43:55.435726 |             | Père Noël  | 1001    |          |
 8530 | ANSWER         | 2012-03-28 13:43:56.672155 |             | Père Noël  | 1001    | 1001     |
 8531 | ANSWER         | 2012-03-28 13:43:56.672554 |             | Tux        | 1000    | 1000     |
 8532 | BRIDGE_START   | 2012-03-28 13:43:56.6726   |             | Tux        | 1000    | 1000     |
 8533 | BRIDGE_END     | 2012-03-28 13:44:01.719856 |             | Tux        | 1000    | 1000     |
 8534 | HANGUP         | 2012-03-28 13:44:01.720438 |             | Père Noël  | 1001    | 1001     |
 8535 | CHAN_END       | 2012-03-28 13:44:01.720481 |             | Père Noël  | 1001    | 1001     |
 8536 | HANGUP         | 2012-03-28 13:44:01.720574 |             | Tux        | 1000    | 1000     |
 8537 | CHAN_END       | 2012-03-28 13:44:01.720985 |             | Tux        | 1000    | 1000     |
(11 rows)
```

1. 1000 calls 1001
2. 1001 *ignores* the call

```
  id  | eventtype  |          eventtime           | userdeftype | cid_name   | cid_num | cid_ani  | c
------+------------+-----------------------------+-------------+------------+---------+----------+---
 8578 | CHAN_START | 2012-03-28 14:02:11.862776 |             | Tux        | 1000    |          |
 8579 | APP_START  | 2012-03-28 14:02:11.924236 |             | Tux        | 1000    | 1000     |
 8580 | CHAN_START | 2012-03-28 14:02:11.924434 |             | Père Noël  | 1001    |          |
 8581 | HANGUP     | 2012-03-28 14:02:21.05216  |             | Père Noël  | 1001    | 1001     |
 8582 | CHAN_END   | 2012-03-28 14:02:21.052217 |             | Père Noël  | 1001    | 1001     |
 8583 | ANSWER     | 2012-03-28 14:02:21.0547   |             | Tux        | 1000    | 1000     |
 8584 | HANGUP     | 2012-03-28 14:02:27.164036 |             | Tux        | 1000    | 1000     |
 8585 | CHAN_END   | 2012-03-28 14:02:27.164529 |             | Tux        | 1000    | 1000     |
(8 rows)
```

Internal Call Using Originate

1. 1000 calls 1001 from his XiVO client
2. 1001 *answers*
3. 1001 *hangs up* after a few seconds

```
  id  |   eventtype    |          eventtime           | userdeftype | cid_name   | cid_num | cid_ani  |
------+----------------+-----------------------------+-------------+------------+---------+----------+
 8538 | CHAN_START     | 2012-03-28 13:46:29.539804 |             | Tux        | 1000    |          |
 8539 | ANSWER         | 2012-03-28 13:46:31.570207 |             | 1001       | 1001    | 1001     |
 8540 | APP_START      | 2012-03-28 13:46:31.633132 |             | Tux        | 1000    | 1001     |
 8541 | CHAN_START     | 2012-03-28 13:46:31.633316 |             | Père Noël  | 1001    |          |
 8542 | ANSWER         | 2012-03-28 13:46:33.356198 |             | Père Noël  | 1001    | 1001     |
 8543 | BRIDGE_START   | 2012-03-28 13:46:33.356388 |             | Tux        | 1000    | 1001     |
 8544 | BRIDGE_END     | 2012-03-28 13:46:39.145872 |             | Tux        | 1000    | 1001     |
 8545 | HANGUP         | 2012-03-28 13:46:39.146556 |             | Père Noël  | 1001    | 1001     |
 8546 | CHAN_END       | 2012-03-28 13:46:39.146633 |             | Père Noël  | 1001    | 1001     |
 8547 | HANGUP         | 2012-03-28 13:46:39.14673  |             | Tux        | 1000    | 1001     |
 8548 | CHAN_END       | 2012-03-28 13:46:39.147224 |             | Tux        | 1000    | 1001     |
(11 rows)
```

1. 1000 calls 1001 from his XiVO client
2. 1001 *ignores* the call

```
  id  | eventtype  |          eventtime           | userdeftype | cid_name   | cid_num | cid_ani  | c
------+------------+-----------------------------+-------------+------------+---------+----------+---
 8594 | CHAN_START | 2012-03-28 14:06:36.616875 |             | Tux        | 1000    |          |
 8595 | ANSWER     | 2012-03-28 14:06:41.370874 |             | 1001       | 1001    | 1001     |
 8596 | APP_START  | 2012-03-28 14:06:41.431579 |             | Tux        | 1000    | 1001     |
```

```
 8597 | CHAN_START | 2012-03-28 14:06:41.431737 |             | Père Noël | 1001     |          |
 8598 | HANGUP     | 2012-03-28 14:06:47.283285 |             | Père Noël | 1001     | 1001     |
 8599 | CHAN_END   | 2012-03-28 14:06:47.283344 |             | Père Noël | 1001     | 1001     |
 8600 | HANGUP     | 2012-03-28 14:06:53.200459 |             | Tux       | 1000     | 1001     |
 8601 | CHAN_END   | 2012-03-28 14:06:53.200924 |             | Tux       | 1000     | 1001     |
(8 rows)
```

1. 1000 calls 1001 from his XiVO client
2. 1000 *ignores* the originate

```
  id  | eventtype  |         eventtime          | userdeftype | cid_name | cid_num | cid_ani  | ci
------+------------+----------------------------+-------------+----------+---------+----------+---
 8602 | CHAN_START | 2012-03-28 14:08:21.083271 |             | Tux      | 1000    |          |
 8603 | HANGUP     | 2012-03-28 14:08:34.431701 |             | 1001     | 1001    | 1001     |
 8604 | CHAN_END   | 2012-03-28 14:08:34.431759 |             | 1001     | 1001    | 1001     |
(3 rows)
```

External Call

1. External user (*555-555-5555*) calls 445
2. 1000 *answers*
3. 1000 *hangs up* after a few seconds

```
  id  |  eventtype   |         eventtime          | userdeftype |  cid_name  |  cid_num   |  cid_
------+--------------+----------------------------+-------------+------------+------------+------
 8567 | CHAN_START   | 2012-03-28 13:58:49.004403 |             | 5555555555 | 5555555555 |
 8568 | APP_START    | 2012-03-28 13:58:49.076665 |             | 5555555555 | 5555555555 | 55555
 8569 | CHAN_START   | 2012-03-28 13:58:49.076842 |             | Tux        | 1000       |
 8570 | ANSWER       | 2012-03-28 13:59:00.173302 |             | Tux        | 1000       | 1000
 8571 | ANSWER       | 2012-03-28 13:59:00.173818 |             | 5555555555 | 5555555555 | 55555
 8572 | BRIDGE_START | 2012-03-28 13:59:00.173872 |             | 5555555555 | 5555555555 | 55555
 8573 | BRIDGE_END   | 2012-03-28 13:59:06.386394 |             | 5555555555 | 5555555555 | 55555
 8574 | HANGUP       | 2012-03-28 13:59:06.38711  |             | Tux        | 1000       | 1000
 8575 | CHAN_END     | 2012-03-28 13:59:06.387153 |             | Tux        | 1000       | 1000
 8576 | HANGUP       | 2012-03-28 13:59:06.387246 |             | 5555555555 | 5555555555 | 55555
 8577 | CHAN_END     | 2012-03-28 13:59:06.387703 |             | 5555555555 | 5555555555 | 55555
(11 rows)
```

1. External user (*555-555-5555*) calls 446
2. 1001 *ignores* the call

```
  id  | eventtype  |         eventtime          | userdeftype |  cid_name  |  cid_num   |  cid_an
------+------------+----------------------------+-------------+------------+------------+--------
 8620 | CHAN_START | 2012-03-28 14:12:13.940952 |             | 5555555555 | 5555555555 |
 8621 | APP_START  | 2012-03-28 14:12:14.028157 |             | 5555555555 | 5555555555 | 5555555
 8622 | CHAN_START | 2012-03-28 14:12:14.02839  |             | Père Noël  | 1001       |
 8623 | HANGUP     | 2012-03-28 14:12:21.070432 |             | Père Noël  | 1001       | 1001
 8624 | CHAN_END   | 2012-03-28 14:12:21.070489 |             | Père Noël  | 1001       | 1001
 8625 | ANSWER     | 2012-03-28 14:12:21.072653 |             | 5555555555 | 5555555555 | 5555555
 8626 | HANGUP     | 2012-03-28 14:12:26.202676 |             | 5555555555 | 5555555555 | 5555555
 8627 | CHAN_END   | 2012-03-28 14:12:26.203237 |             | 5555555555 | 5555555555 | 5555555
(8 rows)
```

Blind Transfer

1. External user (*555-555-5555*) calls 445
2. 1000 *answers*

3. 1000 *blind transfer* to 1001
4. 1001 *answers*
5. 1001 *hangs up* after a few seconds

```
  id  |   eventtype   |          eventtime           | userdeftype |  cid_name  |  cid_num   |  cid_
------+---------------+------------------------------+-------------+------------+------------+-------
 8689 | CHAN_START    | 2012-03-28 14:21:05.928445 |             | 5555555555 | 5555555555 |
 8690 | APP_START     | 2012-03-28 14:21:06.016235 |             | 5555555555 | 5555555555 | 55555
 8691 | CHAN_START    | 2012-03-28 14:21:06.016426 |             | Tux        | 1000       |
 8692 | ANSWER        | 2012-03-28 14:21:07.600329 |             | Tux        | 1000       | 1000
 8693 | ANSWER        | 2012-03-28 14:21:07.600741 |             | 5555555555 | 5555555555 | 55555
 8694 | BRIDGE_START  | 2012-03-28 14:21:07.6008   |             | 5555555555 | 5555555555 | 55555
 8695 | BRIDGE_END    | 2012-03-28 14:21:13.11719  |             | 5555555555 | 5555555555 | 55555
 8696 | HANGUP        | 2012-03-28 14:21:13.117526 |             | Tux        | 1000       | 1000
 8697 | CHAN_END      | 2012-03-28 14:21:13.117574 |             | Tux        | 1000       | 1000
 8698 | APP_START     | 2012-03-28 14:21:13.199251 |             | 5555555555 | 5555555555 | 55555
 8699 | CHAN_START    | 2012-03-28 14:21:13.199432 |             | Père Noël  | 1001       |
 8700 | ANSWER        | 2012-03-28 14:21:16.573668 |             | Père Noël  | 1001       | 1001
 8701 | BRIDGE_START  | 2012-03-28 14:21:16.573876 |             | 5555555555 | 5555555555 | 55555
 8702 | BRIDGE_END    | 2012-03-28 14:21:23.120075 |             | 5555555555 | 5555555555 | 55555
 8703 | HANGUP        | 2012-03-28 14:21:23.120393 |             | Père Noël  | 1001       | 1001
 8704 | CHAN_END      | 2012-03-28 14:21:23.120436 |             | Père Noël  | 1001       | 1001
 8705 | HANGUP        | 2012-03-28 14:21:23.120533 |             | 5555555555 | 5555555555 | 55555
 8706 | CHAN_END      | 2012-03-28 14:21:23.120981 |             | 5555555555 | 5555555555 | 55555
(18 rows)
```

1. External user (*555-555-5555*) calls 445
2. 1000 *answers*
3. 1000 *blind transfer* to 1001 from his XiVO client
4. 1001 *ignores* the call

```
  id  |   eventtype   |          eventtime           | userdeftype |  cid_name  |  cid_num   |  cid_
------+---------------+------------------------------+-------------+------------+------------+-------
 8736 | CHAN_START    | 2012-03-28 14:37:16.228527 |             | 5555555555 | 5555555555 |
 8737 | APP_START     | 2012-03-28 14:37:16.310874 |             | 5555555555 | 5555555555 | 55555
 8738 | CHAN_START    | 2012-03-28 14:37:16.311033 |             | Tux        | 1000       |
 8739 | ANSWER        | 2012-03-28 14:37:18.863805 |             | Tux        | 1000       | 1000
 8740 | ANSWER        | 2012-03-28 14:37:18.863995 |             | 5555555555 | 5555555555 | 55555
 8741 | BRIDGE_START  | 2012-03-28 14:37:18.86402  |             | 5555555555 | 5555555555 | 55555
 8742 | BRIDGE_END    | 2012-03-28 14:37:23.50673  |             | 5555555555 | 5555555555 | 55555
 8743 | HANGUP        | 2012-03-28 14:37:23.506776 |             | Tux        | 1000       | 1000
 8744 | CHAN_END      | 2012-03-28 14:37:23.507286 |             | Tux        | 1000       | 1000
 8745 | APP_START     | 2012-03-28 14:37:23.568572 |             | 5555555555 | 5555555555 | 55555
 8746 | CHAN_START    | 2012-03-28 14:37:23.568773 |             | Père Noël  | 1001       |
 8747 | HANGUP        | 2012-03-28 14:37:28.065358 |             | Père Noël  | 1001       | 1001
 8748 | CHAN_END      | 2012-03-28 14:37:28.065415 |             | Père Noël  | 1001       | 1001
 8749 | ANSWER        | 2012-03-28 14:37:28.067965 |             | 5555555555 | 5555555555 | 55555
 8750 | HANGUP        | 2012-03-28 14:37:33.132613 |             | 5555555555 | 5555555555 | 55555
 8751 | CHAN_END      | 2012-03-28 14:37:33.133324 |             | 5555555555 | 5555555555 | 55555
(16 rows)
```

Attended transfer

1. External user (*555-555-5555*) calls 445
2. 1000 *answers*
3. 1000 initiate an *attended transfer* to 1001
4. 1001 *answers* and talks to 1001

5. 1000 completes the transfer
6. 1001 *hangs up* after a few seconds

```
  id  |  eventtype    |          eventtime          | userdeftype |  cid_name   |  cid_num    |  cid_a
------+---------------+-----------------------------+-------------+-------------+-------------+-------
 8768 | HANGUP        | 2012-03-28 14:52:41.916113 |             | 5555555555 | 5555555555 | 55555
 8769 | CHAN_END      | 2012-03-28 14:52:41.916516 |             | 5555555555 | 5555555555 | 55555
 8770 | CHAN_START    | 2012-03-28 14:52:56.055872 |             | 5555555555 | 5555555555 |
 8771 | APP_START     | 2012-03-28 14:52:56.130032 |             | 5555555555 | 5555555555 | 55555
 8772 | CHAN_START    | 2012-03-28 14:52:56.130213 |             | Tux         | 1000        |
 8773 | ANSWER        | 2012-03-28 14:52:57.701373 |             | Tux         | 1000        | 1000
 8774 | ANSWER        | 2012-03-28 14:52:57.701866 |             | 5555555555 | 5555555555 | 55555
 8775 | BRIDGE_START  | 2012-03-28 14:52:57.701925 |             | 5555555555 | 5555555555 | 55555
 8776 | CHAN_START    | 2012-03-28 14:53:04.604461 |             | Tux         | 1000        |
 8777 | APP_START     | 2012-03-28 14:53:04.665818 |             | Tux         | 1000        | 1000
 8778 | CHAN_START    | 2012-03-28 14:53:04.665996 |             | Père Noël   | 1001        |
 8779 | ANSWER        | 2012-03-28 14:53:06.314275 |             | Père Noël   | 1001        | 1001
 8780 | ANSWER        | 2012-03-28 14:53:06.314717 |             | Tux         | 1000        | 1000
 8781 | BRIDGE_START  | 2012-03-28 14:53:06.314762 |             | Tux         | 1000        | 1000
 8782 | HANGUP        | 2012-03-28 14:53:12.226404 |             | Tux         | 1000        | 1000
 8783 | CHAN_END      | 2012-03-28 14:53:12.226445 |             | Tux         | 1000        | 1000
 8784 | HANGUP        | 2012-03-28 14:53:12.22654  |             | Tux         | 1000        | 1000
 8785 | CHAN_END      | 2012-03-28 14:53:12.226566 |             | Tux         | 1000        | 1000
 8786 | BRIDGE_END    | 2012-03-28 14:53:18.145645 |             | 5555555555 | 5555555555 | 55555
 8787 | HANGUP        | 2012-03-28 14:53:18.146582 |             | Père Noël   | 1001        | 1001
 8788 | CHAN_END      | 2012-03-28 14:53:18.14666  |             | Père Noël   | 1001        | 1001
 8789 | HANGUP        | 2012-03-28 14:53:18.146789 |             | 5555555555 | 5555555555 | 55555
 8790 | CHAN_END      | 2012-03-28 14:53:18.147343 |             | 5555555555 | 5555555555 | 55555
(23 rows)
```

Voice Mail

1. External user (*555-555-55555*) *calls* 445
2. 1000 *ignores* the call
3. External user *leaves* a message
4. External user *hangs up*

```
  id  | eventtype   |          eventtime          | userdeftype |  cid_name   |  cid_num    |  cid_an
------+-------------+-----------------------------+-------------+-------------+-------------+---------
 8814 | CHAN_START | 2012-03-28 15:18:11.270723 |             | 5555555555 | 5555555555 |
 8815 | APP_START  | 2012-03-28 15:18:11.347544 |             | 5555555555 | 5555555555 | 5555555
 8816 | CHAN_START | 2012-03-28 15:18:11.347707 |             | Tux         | 1000        |
 8817 | HANGUP     | 2012-03-28 15:18:11.644857 |             | Tux         | 1000        | 1000
 8818 | CHAN_END   | 2012-03-28 15:18:11.645103 |             | Tux         | 1000        | 1000
 8819 | HANGUP     | 2012-03-28 15:18:11.645858 |             | 5555555555 | 5555555555 | 5555555
 8820 | CHAN_END   | 2012-03-28 15:18:11.645891 |             | 5555555555 | 5555555555 | 5555555
(7 rows)
```

1. 1000 consults his voicemail
2. 1000 presses 1 to hear his message
3. 1000 presses 7 to delete his message
4. 1000 *hangs up*

```
  id  | eventtype   |          eventtime          | userdeftype | cid_name | cid_num | cid_ani | ci
------+-------------+-----------------------------+-------------+----------+---------+---------+---
 8821 | CHAN_START | 2012-03-28 15:24:27.929536 |             | Tux      | 1000    |         |
 8822 | ANSWER     | 2012-03-28 15:24:27.946719 |             | Tux      | 1000    | 1000    |
 8823 | HANGUP     | 2012-03-28 15:25:09.215869 |             | Tux      | 1000    | 1000    |
```

```
 8824 | CHAN_END    | 2012-03-28 15:25:09.215914 |             | Tux      | 1000    | 1000    |
(4 rows)
```

Call Forward

1. 1000 enable unconditional call forwarding to 1001 using **211001*

```
  id  | eventtype   |         eventtime          | userdeftype | cid_name | cid_num | cid_ani | ci
------+-------------+----------------------------+-------------+----------+---------+---------+---
 8825 | CHAN_START  | 2012-03-29 07:48:37.660366 |             | Tux      | 1000    |         |
 8826 | ANSWER      | 2012-03-29 07:48:37.662976 |             | Tux      | 1000    | 1000    |
 8827 | HANGUP      | 2012-03-29 07:48:41.12148  |             | Tux      | 1000    | 1000    |
 8828 | CHAN_END    | 2012-03-29 07:48:41.121864 |             | Tux      | 1000    | 1000    |
(4 rows)
```

1. External user calls 445
2. The call is *forwarded* to 1001
3. 1001 *answers*
4. External user *hangs up* after a few seconds

```
  id  |   eventtype    |         eventtime          |  userdeftype   |  cid_name  |  cid_num    |  c
------+----------------+----------------------------+----------------+------------+-------------+---
 8829 | CHAN_START     | 2012-03-29 07:57:05.918708 |                | 5555555555 | 5555555555 |
 8830 | XIVO_USER_FWD  | 2012-03-29 07:57:05.982490 | XIVO_USER_FWD  | 5555555555 | 5555555555 | 55
 8831 | ANSWER         | 2012-03-29 07:57:05.982497 |                | 5555555555 | 5555555555 | 55
 8832 | APP_START      | 2012-03-29 07:57:10.269364 |                | 5555555555 | 5555555555 | 55
 8833 | CHAN_START     | 2012-03-29 07:57:10.26958  |                | Père Noël  | 1001        |
 8834 | ANSWER         | 2012-03-29 07:57:13.744093 |                | Père Noël  | 1001        | 10
 8835 | BRIDGE_START   | 2012-03-29 07:57:13.744248 |                | 5555555555 | 5555555555 | 55
 8836 | BRIDGE_END     | 2012-03-29 07:57:24.586868 |                | 5555555555 | 5555555555 | 55
 8837 | HANGUP         | 2012-03-29 07:57:24.587584 |                | Père Noël  | 1001        | 10
 8838 | CHAN_END       | 2012-03-29 07:57:24.588184 |                | Père Noël  | 1001        | 10
 8839 | HANGUP         | 2012-03-29 07:57:24.588304 |                | 5555555555 | 5555555555 | 55
 8840 | CHAN_END       | 2012-03-29 07:57:24.588359 |                | 5555555555 | 5555555555 | 55
(11 rows)
```

Call To a Queue

1. External user *calls* 444
2. The call is distributed to queue service
3. Agent 1234 (Père Noël 1001) *answers*
4. External user *hangs up* after a few seconds

```
  id  |   eventtype    |         eventtime          | userdeftype |  cid_name  |  cid_num   |  cid_
------+----------------+----------------------------+-------------+------------+------------+------
 8859 | CHAN_START     | 2012-03-29 08:18:13.983343 |             | 5555555555 | 5555555555 |
 8860 | ANSWER         | 2012-03-29 08:18:14.049882 |             | 5555555555 | 5555555555 | 55555
 8861 | ANSWER         | 2012-03-29 08:18:19.042273 |             | 5555555555 | 5555555555 | 55555
 8862 | APP_START      | 2012-03-29 08:18:20.113057 |             | 5555555555 | 5555555555 | 55555
 8863 | CHAN_START     | 2012-03-29 08:18:20.116129 |             |            |            |
 8864 | CHAN_START     | 2012-03-29 08:18:20.116184 |             |            |            |
 8865 | CHAN_START     | 2012-03-29 08:18:20.11623  |             |            |            |
 8866 | APP_START      | 2012-03-29 08:18:20.213517 |             | 5555555555 | 5555555555 | 55555
 8867 | CHAN_START     | 2012-03-29 08:18:20.213719 |             | Père Noël  | 1001       |
 8868 | ANSWER         | 2012-03-29 08:18:23.174506 |             | Père Noël  | 1001       | 1001
 8869 | ANSWER         | 2012-03-29 08:18:23.174666 |             | 5555555555 | 5555555555 | 55555
```

```
 8870 | BRIDGE_START | 2012-03-29 08:18:23.174713 |             | 5555555555 | 5555555555 | 55555
 8871 | ANSWER       | 2012-03-29 08:18:23.175011 |             | Père Noël  | 1001       |
 8872 | HANGUP       | 2012-03-29 08:18:23.367533 |             | Père Noël  | 1001       | 1001
 8873 | CHAN_END     | 2012-03-29 08:18:23.367547 |             | Père Noël  | 1001       | 1001
 8874 | HANGUP       | 2012-03-29 08:18:23.367592 |             | 5555555555 | 5555555555 | 55555
 8875 | CHAN_END     | 2012-03-29 08:18:23.367604 |             | 5555555555 | 5555555555 | 55555
 8876 | HANGUP       | 2012-03-29 08:18:31.818877 |             |            | 444        |
 8877 | HANGUP       | 2012-03-29 08:18:31.818928 |             | Père Noël  | 1001       |
 8878 | CHAN_END     | 2012-03-29 08:18:31.81938  |             | Père Noël  | 1001       |
 8879 | CHAN_END     | 2012-03-29 08:18:31.819443 |             |            | 444        |
 8880 | HANGUP       | 2012-03-29 08:18:31.823143 |             | 5555555555 | 5555555555 | 55555
 8881 | CHAN_END     | 2012-03-29 08:18:31.823175 |             | 5555555555 | 5555555555 | 55555
(23 rows)
```

Call To the Operator

1. External user *calls* 447
2. The call is routed to the operator (Tux 1000/Agent 9999)
3. The operator *hangs up* after a few seconds

```
  id  |  eventtype   |         eventtime          | userdeftype |  cid_name  |  cid_num   |  cid_a
------+--------------+----------------------------+-------------+------------+------------+-------
 8943 | CHAN_START   | 2012-03-29 08:47:54.875967 |             | 5555555555 | 5555555555 |
 8944 | ANSWER       | 2012-03-29 08:47:54.952265 |             | 5555555555 | 5555555555 | 55555
 8945 | APP_START    | 2012-03-29 08:47:56.050332 |             | 5555555555 | 5555555555 | 55555
 8946 | CHAN_START   | 2012-03-29 08:47:56.05637  |             |            |            |
 8947 | CHAN_START   | 2012-03-29 08:47:56.056423 |             |            |            |
 8948 | CHAN_START   | 2012-03-29 08:47:56.056462 |             |            |            |
 8949 | APP_START    | 2012-03-29 08:47:56.10545  |             | 5555555555 | 5555555555 | 55555
 8950 | CHAN_START   | 2012-03-29 08:47:56.105603 |             | Tux        | 1000       |
 8951 | ANSWER       | 2012-03-29 08:47:59.401063 |             | Tux        | 1000       | 1000
 8952 | ANSWER       | 2012-03-29 08:47:59.401188 |             | 5555555555 | 5555555555 | 55555
 8953 | BRIDGE_START | 2012-03-29 08:47:59.401228 |             | 5555555555 | 5555555555 | 55555
 8954 | ANSWER       | 2012-03-29 08:47:59.40153  |             | Tux        | 1000       |
 8955 | ANSWER       | 2012-03-29 08:47:59.401556 |             |            | 447        |
 8956 | BRIDGE_START | 2012-03-29 08:47:59.417761 |             | 5555555555 | 5555555555 | 55555
 8957 | HANGUP       | 2012-03-29 08:47:59.535268 |             | Tux        | 1000       | 1000
 8958 | CHAN_END     | 2012-03-29 08:47:59.535306 |             | Tux        | 1000       | 1000
 8959 | HANGUP       | 2012-03-29 08:47:59.535397 |             | 5555555555 | 5555555555 | 55555
 8960 | CHAN_END     | 2012-03-29 08:47:59.535425 |             | 5555555555 | 5555555555 | 55555
 8961 | HANGUP       | 2012-03-29 08:48:03.879393 |             | Tux        | 1000       |
 8962 | CHAN_END     | 2012-03-29 08:48:03.87946  |             | Tux        | 1000       |
 8963 | BRIDGE_END   | 2012-03-29 08:48:03.87952  |             | 5555555555 | 5555555555 | 55555
 8964 | HANGUP       | 2012-03-29 08:48:03.882675 |             |            | 447        |
 8965 | CHAN_END     | 2012-03-29 08:48:03.882709 |             |            | 447        |
 8966 | HANGUP       | 2012-03-29 08:48:03.882873 |             | 5555555555 | 5555555555 | 55555
 8967 | CHAN_END     | 2012-03-29 08:48:03.883324 |             | 5555555555 | 5555555555 | 55555
(25 rows)
```

1. External user *calls* 447
2. The call is routed to the operator (Tux 1000/Agent 9999)
3. The operator *transfers* the call to 1001

```
  id  |  eventtype   |         eventtime          | userdeftype |  cid_name  |  cid_num   |  cid_a
------+--------------+----------------------------+-------------+------------+------------+-------
 8911 | CHAN_START   | 2012-03-29 08:41:14.215721 |             | 5555555555 | 5555555555 |
 8912 | ANSWER       | 2012-03-29 08:41:14.301682 |             | 5555555555 | 5555555555 | 55555
 8913 | APP_START    | 2012-03-29 08:41:15.37998  |             | 5555555555 | 5555555555 | 55555
 8914 | CHAN_START   | 2012-03-29 08:41:15.386099 |             |            |            |
```

```
 8915 | CHAN_START   | 2012-03-29 08:41:15.38615  |              |            |            |
 8916 | CHAN_START   | 2012-03-29 08:41:15.386214 |              |            |            |
 8917 | APP_START    | 2012-03-29 08:41:15.449043 |              | 5555555555 | 5555555555 | 55555
 8918 | CHAN_START   | 2012-03-29 08:41:15.449351 |              | Tux        | 1000       |
 8919 | ANSWER       | 2012-03-29 08:41:17.624003 |              | Tux        | 1000       | 1000
 8920 | ANSWER       | 2012-03-29 08:41:17.624122 |              | 5555555555 | 5555555555 | 55555
 8921 | BRIDGE_START | 2012-03-29 08:41:17.624159 |              | 5555555555 | 5555555555 | 55555
 8922 | ANSWER       | 2012-03-29 08:41:17.624454 |              | Tux        | 1000       |
 8923 | ANSWER       | 2012-03-29 08:41:17.62448  |              |            | 447        |
 8924 | BRIDGE_START | 2012-03-29 08:41:17.632978 |              | 5555555555 | 5555555555 | 55555
 8925 | HANGUP       | 2012-03-29 08:41:17.767612 |              | Tux        | 1000       | 1000
 8926 | CHAN_END     | 2012-03-29 08:41:17.767645 |              | Tux        | 1000       | 1000
 8927 | HANGUP       | 2012-03-29 08:41:17.767733 |              | 5555555555 | 5555555555 | 55555
 8928 | CHAN_END     | 2012-03-29 08:41:17.76776  |              | 5555555555 | 5555555555 | 55555
 8929 | BRIDGE_END   | 2012-03-29 08:41:22.12071  |              | 5555555555 | 5555555555 | 55555
 8930 | HANGUP       | 2012-03-29 08:41:22.125079 |              |            | 447        |
 8931 | HANGUP       | 2012-03-29 08:41:22.125103 |              | Tux        | 1000       |
 8932 | CHAN_END     | 2012-03-29 08:41:22.125151 |              | Tux        | 1000       |
 8933 | CHAN_END     | 2012-03-29 08:41:22.125209 |              |            | 447        |
 8934 | APP_START    | 2012-03-29 08:41:22.194926 |              | 5555555555 | 5555555555 | 55555
 8935 | CHAN_START   | 2012-03-29 08:41:22.195135 |              | Père Noël  | 1001       |
 8936 | ANSWER       | 2012-03-29 08:41:25.236852 |              | Père Noël  | 1001       | 1001
 8937 | BRIDGE_START | 2012-03-29 08:41:25.237116 |              | 5555555555 | 5555555555 | 55555
 8938 | BRIDGE_END   | 2012-03-29 08:41:25.829818 |              | 5555555555 | 5555555555 | 55555
 8939 | HANGUP       | 2012-03-29 08:41:25.830079 |              | Père Noël  | 1001       | 1001
 8940 | CHAN_END     | 2012-03-29 08:41:25.830118 |              | Père Noël  | 1001       | 1001
 8941 | HANGUP       | 2012-03-29 08:41:25.830213 |              | 5555555555 | 5555555555 | 55555
 8942 | CHAN_END     | 2012-03-29 08:41:25.83061  |              | 5555555555 | 5555555555 | 55555
(32 rows)
```

Userfield with CEL

The CEL table has a field named *userfield* which is used on outgoing calls when the caller has a userfield set in his configuration. The *userfield* can be enriched from the dialplan with context information.

To set the value of the field from the dialplan you have to you the asterisk function *CHANNEL(userfield)*, for example:

```
Set(CHANNEL(userfield)=${MY_DIALPLAN_VAR})
```

Note : for those who used the same *userfield* information in the CDR by calling the function *CDR(userfield)*, you have to replace it with the function *CHANNEL(userfield)*.

1.11.2 Message Bus

The message bus is used to receive events from XiVO. It is provided by an AMQP 0-9-1 broker (namely, RabbitMQ) that is integrated in XiVO.

> **Warning:** Interaction with the bus is presently experimental and some things might change in the next XiVO versions.

Usage

At the moment, the AMQP broker only listen on the 127.0.0.1 address. This means that if you want to connect to the AMQP broker from a distant machine, you must modify the RabbitMQ server configuration, which is not yet an officially supported operation.

Otherwise, the default connection information is:

- Virtual host: /
- User name: guest
- User password: guest
- Port: 5672

Example

Here's an example of a simple client, in python, listening for the *call_form_result* CTI events:

```
#!/usr/bin/python

import pika

connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
channel = connection.channel()

result = channel.queue_declare(exclusive=True)
queue_name = result.method.queue

channel.queue_bind(exchange='xivo-cti', queue=queue_name, routing_key='call_form_result')

def callback(ch, method, props, body):
    print 'Received:', body
    ch.basic_ack(delivery_tag=method.delivery_tag)

channel.basic_consume(callback, queue=queue_name)
channel.start_consuming()
```

If you are new to AMQP, you might want to look at the RabbitMQ tutorial.

Notes

Things to be aware when writing a client/consumer:

- The `xivo-service stop` command stops the AMQP broker. This means that the client connections to the AMQP broker will be lost on:
  - a XiVO upgrade
  - an asterisk crash
- The published messages are not persistant. When the AMQP broker stops, the messages that are still in queues will be lost.

Events

Events that are sent to the bus use a JSON serialization format. For example, the CTI call_form_result event looks like this:

```
{"name": "call_form_result", "data": {...}}
```

All events have the same basic structure, namely, a JSON object with two keys:

name A string representing the name of the event. Each event type has a unique name.

data The data specific part of the event. This is documented on a per event type; if not this is assumed to be null.

AMI

AMI related events are sent to the `xivo-ami` exchange, which is an exchange of type topic.

To subscribe to event with name X, you must create a binding between the exchange and your queue with the binding/routing key X.

Example event with binding key QueueMemberStatus:

```
{
    "name": "QueueMemberStatus",
    "data": {
        "Status": "1",
        "Penalty": "0",
        "CallsTaken": "0",
        "Skills": "",
        "MemberName": "sip\/m3ylhs",
        "Queue": "petak",
        "LastCall": "0",
        "Membership": "static",
        "Location": "sip\/m3ylhs",
        "Privilege": "agent,all",
        "Paused": "0",
        "StateInterface": "sip\/m4ylhs"
    }
}
```

CTI

CTI related events are sent to the `xivo-cti` exchange, which is an exchange of type direct.

To subscribe to event with name X, you must create a binding between the exchange and your queue with the binding/routing key X.

call_form_result The call_form_result event is sent when a *custom call form* is submitted by a CTI client.

- routing key: call_form_result
- event specific data: a dictionary with 2 keys:
  - user_id: an integer corresponding to the user ID of the client who saved the call form
  - variables: a dictionary holding the content of the form

Example:

```
{
    "name": "call_form_result",
    "data": {
        "user_id": 40,
        "variables": {
            "firstname": "John",
            "lastname": "Doe"
        }
    }
}
```

1.11.3 Queue logs

Queue logs are events logged by Asterisk in the queue_log table of the asterisk database. Queue logs are used to generate XiVO call center statistics.

Queue log sample

Agent callback login

```
           time             |      callid      | queuename |   agent    |         event         |
----------------------------+------------------+-----------+------------+-----------------------+----
2012-07-03 15:27:23.896208 | 1341343640.4     | NONE      | Agent/3001 | AGENTCALLBACKLOGIN   | 100
```

Agent callback logoff

Agent/3001 is logged in queues q1 and q2.

```
           time             |      callid      | queuename |   agent    |         event         |
----------------------------+------------------+-----------+------------+-----------------------+----
2012-07-03 15:28:07.348244 | NONE             | q2        | Agent/3001 | UNPAUSE               |
2012-07-03 15:28:07.346320 | NONE             | q1        | Agent/3001 | UNPAUSE               |
2012-07-03 15:28:07.327425 | NONE             | NONE      | Agent/3001 | UNPAUSEALL            |
2012-07-03 15:28:06.249357 | NONE             | NONE      | Agent/3001 | AGENTCALLBACKLOGOFF  | 100
```

Call on a Queue with join empty conditions met

```
           time             |      callid      | queuename |   agent    |         event         |
----------------------------+------------------+-----------+------------+-----------------------+----
2012-07-04 07:27:55.640421 | 1341401275.9     | q1        | NONE       | JOINEMPTY             |
```

Enter the queue and get answered by an agent

```
           time             |      callid      | queuename |   agent    |         event         |
----------------------------+------------------+-----------+------------+-----------------------+----
2012-07-04 07:33:23.085718 | 1341401601.24    | q1        | Agent/3001 | CONNECT               | 2
2012-07-04 07:33:21.165823 | 1341401601.24    | q1        | NONE       | ENTERQUEUE            |
```

Agent or caller ends the call after 12 seconds

```
           time             |      callid      | queuename |   agent    |         event         |
----------------------------+------------------+-----------+------------+-----------------------+----
2012-07-04 07:37:46.601754 | 1341401851.34    | q1        | Agent/3001 | COMPLETEAGENT         | 2
```

Call on a full queue

```
           time             |      callid      | queuename |   agent    |         event         |
----------------------------+------------------+-----------+------------+-----------------------+----
2012-07-04 07:40:17.339945 | 1341402016.44    | q1        | NONE       | FULL                  |
```

Call on a closed queue

```
           time             |      callid      | queuename |   agent    |         event         |
----------------------------+------------------+-----------+------------+-----------------------+----
2012-07-04 07:48:03.455999 | 1341402482.49    | q1        | NONE       | CLOSED                |
```

Caller abandon before an answer

```
            time              |      callid      | queuename |   agent     |         event         |
----------------------------+------------------+-----------+-------------+-----------------------+---
 2012-07-04 07:49:52.939802 | 1341402586.51    | q1        | NONE        | ABANDON               | 1
```

1.11.4 REST API

The XiVO REST API is the privileged way to programmatically interact with XiVO.

The current API version is *1.1*.

Configuration

The REST API is available via HTTPS on port 50051. Accessing the REST API requires to create a webservices user in the web interface (Configuration/Management/Web Services Access):

- if an IP address is specified for the user, no authentication is needed
- if you choose not to specify an IP address for the user, you can connect to the REST API with a HTTP Digest authentication, using the user name and password you provided. For instance, the following command line allows to retrieve XiVO users through the REST API, using the login **admin** and the password **passadmin**:

  ```
  curl --digest --insecure -u admin:passadmin https://<xivo_address>:50051/1.1/users
  ```

The REST API is also available on the loopback interface via HTTP on port 50050, with no authentication needed.

HTTP status codes

Standard HTTP status codes are used. For the full definition see IANA definition.

- 200: Success
- 201: Created
- 400: Incorrect syntax
- 404: Resource not found
- 406: Not acceptable
- 412: Precondition failed
- 415: Unsupported media type
- 500: Internal server error

General URL parameters

All URL's starts by /1.1/, 1.1 is the current protocol version.

Example usage of general parameters:

```
GET http://127.0.0.1:50050/1.1/voicemails?limit=X&skip=Y
```

Parameters

order Sort the list using a column (e.g. "number"). See specific resource documentation for columns allowed.

direction 'asc' or 'desc'. Sort list in ascending (asc) or descending (desc) order

limit total number of resources to show in the list. Must be a positive integer

skip number of resources to skip over before starting the list. Must be a positive integer

search Search resources. Only resources with a field containing the search term will be listed.

Data representation

Data retrieved from the REST server

JSON is used to encode returned or sent data. Therefore, the following headers are needed:

- **when the request is supposed to return JSON:** Accept = application/json
- **when the request's body contains JSON:** Content-Type = application/json

Note: Optional properties can be added without changing the protocol version in the main list or in the object list itself. Properties will not be removed, type and name will not be modified.

Getting object lists `GET /1.1/objects`

When returning lists the format is as follows:

- total - number of items in total in the system configuration (optional)
- items - returned data as an array of object properties list.

Other optional properties can be added later.

`Response data format`

```
{
    "total": 2,

    "items":
    [
        {
            "id": "1",
            "prop1": "test"
        },
        {
            "id": "2",
            "prop1": "ssd"
        }
    ]
}
```

Getting An Object Format returned is a list of properties. The object should always have the same attributes set, the default value being the equivalent to NULL in the content-type format.

`GET /1.1/objects/<id>`

`Response data format`

```
{
   "id": "1",
   "prop1": "test"
}
```

Data sent to the REST server

The XiVO REST server implements POST and PUT methods for item creation and update respectively. Data is created using the POST method via a root URL and is updated using the PUT method via a root URL suffixed by /<id. The server expects to receive JSON encoded data. Only one item can be processed per request. The data format and required data fields are illustrated in the following example:

```
Request data format
```

```
{
   "id": "1",
   "prop1": "test"
}
```

When updating, only the id and updated properties are needed, omitted properties are not updated. Some properties can also be optional when creating an object.

Errors

A request to the web services may return an error. An error will always be associated to an HTTP error code, and eventually to one or more error messages. The following errors are common to all web services:

Error code	Error message	Description
406	empty	Accept header missing or contains an unsupported content type
415	empty	Content-Type header missing or contains an unsupported content type
500	list of errors	An error occured on the server side; the content of the message depends of the type of errors which occured

The 400, 404 and 412 errors depend on the web service you are requesting. They are separately described for each of them.

The error messages are contained in a JSON list, even if there is only one error message:

```
[ message_1, message_2, ... ]
```

API

Beta version:

REST API 1.1

Note: REST API 1.1 is currently evolving. New features and small fixes are regularly being added over time. We invite the reader to periodically check the changelog for an update on new features and changes.

Warning: Some services are still being developped and can be changed without prior warning. Use at your own risk. Here is a list of services in BETA stage:

- Function Keys
- Line Extension Associations

Call Logs

Call Logs Representation

Description

Field	Values	Description
Call date	date	YYYY-MM-DDTHH:MM:SS
Caller	string	
Called	string	
Period	integer	Number of seconds of the call. 0 if not answered
user Field	string	

Example

```
Call Date,Caller,Called,Period,user Field
2013-01-02T00:00:00,source2 (1002),2002,2,userfield
```

Format Call logs are presented in CSV format, with the following specifications:

- field names are listed on the first line
- fields are separated by commas: `,`
- if there is a comma in a field value, the value is surrounded by double quotes: `"`
- the CSV file uses the character encoding UTF-8

List Call logs

Note: Call logs are generated automatically, but not immediately. See *Call Logs*.

Query

```
GET /1.1/call_logs
```

Example request

```
GET /1.1/call_logs HTTP/1.1
Host: xivoserver
Accept: text/csv
```

Example response

```
HTTP/1.1 200 OK
Content-Type: text/csv; charset=utf8

Call Date,Caller,Called,Period,user Field
2013-01-01T00:00:00,source1 (1001),2001,1,
2013-01-02T00:00:00,source2 (1002),2002,2,userfield
```

Filtering by period

Query

```
GET /1.1/call_logs?start_date=DATE&end_date=DATE
```

`DATE` must be in the following format: `YYYY-MM-DDTHH:MM:SS`. Note the `T` separating the date and time. `start_date` and `end_date` must be given together ; the REST API will not accept `start_date` without `end_date` and vice-versa.

Example request

```
GET /1.1/call_logs?start_date=2013-01-01T00:12:34&end_date=2013-01-02T06:54:32 HTTP/1.1
Host: xivoserver
Accept: text/csv
```

Example response

```
HTTP/1.1 200 OK
Content-Type: text/csv; charset=utf8

Call Date,Caller,Called,Period,user Field
2013-01-01T01:00:00,source1 (1001),2001,1,
2013-01-02T00:00:00,source2 (1002),2002,2,userfield
```

CTI Profiles

CTI Profiles representation

Description

Field	Values	Description
id	integer	Read-only
name	string	Display name

Example

```
{
    "id": 1,
    "name": "Client"
}
```

CTI Profiles list

Query

```
GET /1.1/cti_profiles
```

Example requests Listing all available CTI profiles:

```
GET /1.1/cti_profiles HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "total": 2,
    "items":
    [
        {
             "id": 1,
             "name": "Client"
        },
        {
             "id": 2,
             "name": "Agent"
        }
    ]
}
```

Get CTI Profile

Query

```
GET /1.1/cti_profiles/<id>
```

Example request

```
GET /1.1/cti_profiles/1 HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
     "id": 1,
     "name": "Client"
}
```

Devices

Device Representation

Description

Field	Values	Description
id	string	(Read-only)
ip	string formatted as an IP address (10.0.0.0)	IP address
mac	string formatted as a MAC address (aa:bb:cc:dd:ee:ff)	MAC address
sn	string	Serial number
vendor	string	Vendor name
model	string	Device model
version	string	Firmware version
plugin	string	Provisioning plugin to be used by the device
description	string	
status	• configured • autoprov • not_configured	• configured: Device is configured and ready to be used • autoprov : Device can be provisionned using a provisioning code • not_configured : Device has not been completely configured
options	object	List of standard keys: • switchboard : a boolean indicating if this device is a switchboard
template_id	string	ID of the device template. All devices using a device template will have a certain number of common parameters preconfigured for the device

Example

```
{
    "id": "412c212cff500cc158f373ff00e078f7",
    "ip": "10.0.0.1",
    "mac": "00:00:5e:00:00:01",
    "sn": null,
    "vendor": "Aastra",
    "model": "6757i",
    "version": "3.2.2",
    "plugin": "xivo-aastra-3.2.2-SP3",
    "description": null,
    "status": "configured",
    "options": {"switchboard": true},
    "template_id": "defaultconfigdevice",
    "links" : [
        {
            "rel": "devices",
            "href": "https://xivoserver/1.1/devices/412c212cff500cc158f373ff00e078f7"
        }
    ]
}
```

Device list

Query

```
GET /1.1/devices
```

Parameters

order Sort devices using the specified field (e.g. "mac"). Allowed fields: ip, mac, plugin, model, vendor, version.

direction 'asc' or 'desc'. Sort list in ascending (asc) or descending (desc) order

limit total number of devices to show in the list

skip number of devices to skip over before starting the list

search Search devices. Only devices with a field containing the search term will be listed.

Errors

Error code	Error message	Description
400	Invalid parameters: limit must be a positive number	the 'limit' parameter must be a number
400	Invalid parameters: skip must be a positive number	the 'skip' parameter must be a number
400	Invalid parameters: ordering parameter '<field>' does not exist	you must use one of the fields available in a device when sorting a list
400	Invalid parameters: direction parameter '<direction>' does not exist	use either 'asc' or 'desc' as a direction when sorting a list

Example requests List all devices:

```
GET /1.1/devices HTTP/1.1
Host: xivoserver
Accept: application/json
```

List 10 devices, sorted by mac address:

```
GET /1.1/devices?limit=10&order=mac HTTP/1.1
Host: xivoserver
Accept: application/json
```

Search for devices containing the term “aastra”:

```
GET /1.1/devices?search=aastra HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "total": 2,
    "items": [
        {
            "id": "412c212cff500cc158f373ff00e078f7",
            "ip": "10.0.0.1",
            "mac": "00:00:5e:00:00:01",
            "sn": null,
            "vendor": "Aastra",
            "model": "6731i",
            "version": "3.2.2",
            "plugin": "xivo-aastra-3.2.2-SP3"
            "description": null,
            "status": "configured",
            "options": null,
            "template_id": "defaultconfigdevice",
            "links" : [
                {
                    "rel": "devices",
                    "href": "https://xivoserver/1.1/devices/412c212cff500cc158f373ff00e078f7"
                }
            ]
        },
        {
            "id": "6ff76e09a7ab51ec3afe152a63324ff9",
            "ip": "10.0.0.2",
            "mac": "00:00:5e:00:00:02",
            "sn": null,
            "vendor": "Snom",
            "model": "720",
            "version": "8.7.3.19",
            "plugin": "xivo-snom-8.7.3.19",
            "description": null,
            "status": "configured",
            "options": null,
            "template_id": "defaultconfigdevice",
            "links" : [
                {
                    "rel": "devices",
                    "href": "https://xivoserver/1.1/devices/6ff76e09a7ab51ec3afe152a63324ff9"
                }
            ]
        }
    ]
}
```

Get Device

Query

```
GET /1.1/devices/<id>
```

Parameters

id Device's id

Errors

Error code	Error message	Description
404	Not found	The requested device was not found

Example request

```
GET /1.1/devices/412c212cff500cc158f373ff00e078f7 HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": "412c212cff500cc158f373ff00e078f7",
    "ip": "10.0.0.1",
    "mac": "00:00:5e:00:00:01",
    "sn": null,
    "vendor": "Aastra",
    "model": "6731i",
    "version": "3.2.2",
    "plugin": "xivo-aastra-3.2.2-SP3"
    "description": null,
    "status": "configured",
    "options": null,
    "template_id": "defaultconfigdevice",
    "links" : [
        {
            "rel": "devices",
            "href": "https://xivoserver/1.1/devices/412c212cff500cc158f373ff00e078f7"
        }
    ]
}
```

Create a Device

Query

```
POST /1.1/devices
```

Input

Field	Required	Values	Description
ip	no	string	(see above)
mac	no	string	(see above)
sn	no	string	(see above)
vendor	no	string	(see above)
model	no	string	(see above)
version	no	string	(see above)
description	no	string	(see above)
options	no	object	(see above)
template_id	no	string	(see above)

Errors

Error code	Error message	Description
400	error while creating Device: <explanation>	See explanation for more details
400	Invalid parameters: ip	ip address is not formatted correctly
400	Invalid parameters: mac	mac address is not formatted correctly
400	Invalid parameters: options	options is not an object
400	Invalid parameters: options.switchboard	switchboard option is not a boolean
400	device <mac> already exists	a device using the same MAC address has already been created
400	Nonexistent parameters: plugin <plugin> does not exist	the selected plugin does not exist or has not been installed
400	Nonexistent parameters: template_id <template_id> does not exist	the selected device template does not exist

Example request

```
POST /1.1/devices HTTP/1.1
Host: xivoserver
Accept: application/json
Content-Type: application/json

{
    "ip": "10.0.0.1",
    "mac": "00:00:5e:00:00:01",
    "vendor": "Aastra",
    "model": "6731i",
    "version": "3.2.2",
    "plugin": "xivo-aastra-3.2.2-SP3"
    "template_id": "defaultconfigdevice",
}
```

Example response

```
HTTP/1.1 201 Created
Location: /1.1/devices/412c212cff500cc158f373ff00e078f7
Content-Type: application/json

{
    "id": "412c212cff500cc158f373ff00e078f7",
    "ip": "10.0.0.1",
    "mac": "00:00:5e:00:00:01",
    "sn": null,
    "vendor": "Aastra",
    "model": "6731i",
    "version": "3.2.2",
    "description": null,
    "status": "configured",
    "plugin": "xivo-aastra-3.2.2-SP3"
    "options": null,
    "template_id": "defaultconfigdevice",
    "links" : [
        {
            "rel": "devices",
            "href": "https://xivoserver/1.1/devices/412c212cff500cc158f373ff00e078f7"
        }
    ]
}
```

Update a Device

Query

```
PUT /1.1/devices/<id>
```

The update does not need to set all the fields for the device. Only the fields that need to be updated must be set.

Parameters

id Device's id

Input Same as for creating a device. Please see Create a Device

Errors Same as for creating a device. Please see Create a Device

Example request

```
PUT /1.1/devices/42 HTTP/1.1
Host: xivoserver
Content-Type: application/json

{
    "ip": "10.0.0.1"
}
```

Example response

```
HTTP/1.1 204 No Content
```

Delete a Device A device can not be deleted if it is linked to a line. You must deassociate the line and the device first.

Query

```
DELETE /1.1/devices/<id>
```

Errors

Error code	Error message	Description
400	error while deleting Device: <explanation>	See error message for more details
404	Not found	The requested device was not found

Example request

```
DELETE /1.1/devices/412c212cff500cc158f373ff00e078f7 HTTP/1.1
Host: xivoserver
```

Example response

```
HTTP/1.1 204 No Content
```

Reset a device to autoprov

Warning: The device's configuration will be lost when reset to autoprov mode.

Resets a device into 'autoprov' mode. Once in autoprov, a device can be reprovisionned using another provisioning code.

Query

```
GET /1.1/devices/<id>/autoprov
```

Parameters

id Device's id

Example request

```
GET /1.1/devices/412c212cff500cc158f373ff00e078f7/autoprov
Host: xivoserver
```

Example response

```
HTTP/1.1 204 No Content
```

Synchronize a device Synchronize a device's configuration. Used when a configuration has been modified and the changes need to be sent to the device.

Query

```
GET /1.1/devices/<id>/synchronize
```

Parameters

id Device's id

Example request

```
GET /1.1/devices/412c212cff500cc158f373ff00e078f7/synchronize
Host: xivoserver
```

Example response

```
HTTP/1.1 204 No Content
```

Associate a line to a device

Warning: This feature is not yet accessible nor functional.

After associating a line, the device needs to be synchronized for the changes to take effect. Please see Synchronize a device

Query

```
GET /1.1/devices/<id>/associate_line/<lineid>
```

Parameters

id Device's id

line_id Line id

Example request

```
GET /1.1/devices/412c212cff500cc158f373ff00e078f7/associate_line/2
Host: xivoserver
```

Example response

```
HTTP/1.1 204 No Content
```

Remove a line from a device

Warning: This feature is not yet accessible nor functional.

After removing a line, the device needs to be synchronized for the changes to take effect. Please see Synchronize a device

Query

```
GET /1.1/devices/<id>/remove_line/<lineid>
```

Parameters

id Device's id

line_id Line id

Example request

```
GET /1.1/devices/412c212cff500cc158f373ff00e078f7/remove_line/2
Host: xivoserver
```

Example response

```
HTTP/1.1 204 No Content
```

Extensions

An extension represents a number that can be dialed on a phone. Once an extension is created, it can be associated with different kinds of resources. These associations determine where a call will be routed when the extension is dialed.

An extension is composed of an "exten" (the number to dial) and a "context" (from where are we allowed to dial). The context restrains what source a call will come in from. (e.g. DID calls will come from the context "from-extern")

Extension Representation

Description

Field	Values	Description
id	int	Read-only
exten	string	
context	string	
commented	boolean	If True the extension is disabled
links	list	The link to the resource

Example

```
{
    "id": 1,
    "context": "default",
    "exten": "1234",
    "commented": false,
    "links" : [
        {
            "rel": "extensions",
            "href": "https://xivoserver/1.1/extensions/1"
        }
```

```
    ]
}
```

List Extension

Query

```
GET /1.1/extensions
```

Example request

```
GET /1.1/extensions HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "total": 2,
    "items": [
        {
            "id": 1,
            "context": "default",
            "exten": "1234",
            "commented": false,
            "links" : [
                {
                    "rel": "extensions",
                    "href": "https://xivoserver/1.1/extensions/1"
                }
            ]
        },
        {
            "id": 2,
            "context": "default",
            "exten": "6789",
            "commented": true,
            "links" : [
                {
                    "rel": "extensions",
                    "href": "https://xivoserver/1.1/extensions/2"
                }
            ]
        }
    ]
}
```

Get Extension

Query

```
GET /1.1/extensions/<id>
```

Example request

```
GET /1.1/extensions/1 HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 1,
    "context": "default",
    "exten": "1234",
    "commented": false
}
```

Create Extension The extension number must be included in one of the extension ranges for the given context.

Query

```
POST /1.1/extensions
```

Input

Field	Required	Values	Description
exten	yes	string	
context	yes	string	
commented	no	boolean	If True the extension is disabled.

Errors

Error code	Error message	Description
400	exten <number> not inside range of context <context>	
400	error while creating Extension: <explanation>	See error message for more details

Example request

```
POST /1.1/extensions HTTP/1.1
Host: xivoserver
Accept: application/json
Content-Type: application/json

{
    "exten": "1234",
    "context": "default",
    "commented": false
}
```

Example response

```
HTTP/1.1 201 Created
Location: /1.1/extensions/1
Content-Type: application/json

{
    "id": 1,
    "links" : [
        {
            "rel": "extensions",
            "href": "https://xivoserver/1.1/extensions/1"
        }
```

```
    ]
}
```

Update an Extension The update does not need to set all the fields of the edited extension. The update only needs to set the modified fields. The new extension number must be included in one of the extension ranges for the new context.

Query

```
PUT /1.1/extensions/<id>
```

Errors

Error code	Error message	Description
400	error while editing Extension: <explanation>	See error message for more details
400	exten <number> not inside range of <context>	
404	Not found	The requested extension was not found

Example request

```
PUT /1.1/extensions/42 HTTP/1.1
Host: xivoserver
Content-Type: application/json

{
    "context": "my_context"
}
```

Example response

```
HTTP/1.1 204 No Content
```

Delete Extension An extension can not be deleted if it is associated to a line. You must delete the association first. Consult the documentation on *Line Extension Association* for further details.

Query

```
DELETE /1.1/extensions/<id>
```

Errors

Error code	Error message	Description
400	error while deleting Extension: <explanation>	See error message for more details
400	Error while deleting Extension: extension still has a link	See explanation above
404	Not found	The requested extension was not found

Example request

```
DELETE /1.1/extensions/1 HTTP/1.1
Host: xivoserver
```

Example response

```
HTTP/1.1 204 No Content
```

Line-Extension Association See *Line Extension Association*.

Function Keys Service for configuring what a function key will do when pressed. A function key can accomplish different actions depending on their type and destination.

This service does not add a function key to a device. Consult the documentation on function key templates for further details.

Warning: The function key template service has not been implemented yet

Function Key Representation

Description

Field	Values	Description
id	integer	(Read-only)
type	string	See Function Key Types for more details
destination	string	See Function Key destinations for speed dials for more details
destination_id	integer	See Function Key destinations for speed dials for more details

Example

```
{
    "id": "1",
    "type": "speeddial",
    "destination": "user",
    "destination_id": 34,
    "links": [
         {
             "rel": "func_keys",
             "href: "https://xivoserver/1.1/func_keys/1"
         }
     ]
 }
```

Function Key Types A type determines what kind of action a function key can accomplish. Here is a list of available types and what action they trigger:

speeddial Call another extension. (e.g. a user, a queue, a group, etc).

transfer Transfer the current call to another extension.

dtmf Emit a dial tone, as if you pressed a number on the dialpad (e.g. 3, 5, 6).

line Key for accessing the phone's line (i.e. emit or answer calls by pressing on the key).

directory Search for a contact in the phone directory.

park Park the current call in a parking lot.

dnd Activate Do-Not-Disturb on the phone.

app_services Start a custom application on the phone.

Function Key destinations for speed dials A destination determines the number to dial when using 'speeddial' function keys. Destinations are configured by specifying the type of destination and its id. Destinations are pre-generated every time a new resource is created. In other words, a new destination will appear in the List of Function Key Destinations every time a user, group, queue, etc is created. Therefore, function keys do not need to be created manually. However, there is an exception to this rule: Function keys of type 'custom' cannot be pre-generated because the user must manually enter the number to dial.

Here is a list of available destination types:

user A User

group Conference group

queue Calling queue

conference Conference room

custom A custom number to dial, defined by the user

bs_filter Boss/Secretary filter

voicemail A voicemail

paging Paging

Example To configure a function key that would dial the extension of user "Bob" (who has the id 12), you would have a "destination" of type "user" with the "destination_id" 12.

Here is an example of the JSON representation for this user:

```
{
    "type": "speeddial",
    "destination": "user",
    "destination_id": 12
}
```

List of Function Key Destinations

Query

```
GET /1.1/func_keys
```

Parameters

order Sort the list using a column (e.g. "destination"). Columns allowed: type, destination

direction 'asc' or 'desc'. Sort list in ascending (asc) or descending (desc) order

limit total number of function keys to show in the list. Must be a positive integer

skip number of function keys to skip over before starting the list. Must be a positive integer

search Search function keys. Only function keys with a field containing the search term will be listed.

Errors

Error code	Error message	Description
400	Invalid parameters: limit must be a positive number	the 'limit' parameter must be a number
400	Invalid parameters: skip must be a positive number	the 'skip' parameter must be a number
400	Invalid parameters: ordering parameter '<field>' does not exist	you must use one of the fields available in a device when sorting a list
400	Invalid parameters: direction parameter '<direction>' does not exist	use either 'asc' or 'desc' as a direction when sorting a list

Example requests List all available function key destinations:

```
GET /1.1/func_keys HTTP/1.1
Host: xivoserver
Accept: application/json
```

List function key destinations, sort by destination in descending order:

```
GET /1.1/func_keys?order=destination&direction=desc
Host: xivoserver
Accept: application/json
```

List only the first 10 function key destinations containing the word “user”:

```
GET /1.1/func_keys?search=user&limit=10
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "total": 2,
    "items": [
         {
             "id": "1",
             "type": "speeddial",
             "destination": "user",
             "destination_id": 12,
             "links": [
                 {
                     "rel": "func_keys",
                     "href: "https://xivoserver/1.1/func_keys/1"
                 }
             ]
         },
         {
             "id": "2",
             "type": "transfer",
             "destination": "queue",
             "destination_id": 24,
             "links": [
                 {
                     "rel": "func_keys",
                     "href: "https://xivoserver/1.1/func_keys/2"
                 }
             ]
         }
     ]
 }
```

Get a Function Key Destination

Query

```
GET /1.1/func_keys/<id>
```

Example request

```
GET /1.1/func_keys/1 HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

 {
     "id": "1",
     "type": "speeddial",
     "destination": "user",
     "destination_id": 12,
     "links": [
         {
             "rel": "func_keys",
             "href: "https://xivoserver/1.1/func_keys/2"
         }
     ]
 }
```

Create a Function Key Destination Most function keys are automatically generated upon the creation of a desintation resource (See Function Key destinations for speed dials for further details). This action is for creating function keys that cannot be pre-generated (i.e. custom speed dials and other types of function keys)

> **Warning:** Not implemented yet

Query

`POST /1.1/func_keys`

Input

Field	Required	Values	Notes
type	yes	string	See Function Key Types for more details
destination	yes	string	See Function Key destinations for speed dials for more details
destination_id	yes	integer	destination's id

Errors

Error code	Error message	Description
500	Error while creating Function Key: <explanation>	See explanation for more details.
400	Missing parameters: <list of missing fields>	
400	Invalid parameters: type <type> does not exist	Please use one of the function key types listed in Function Key Types
400	Invalid parameters: destination of type <type> does not exist	Please use one of the destination types listed in Function Key destinations for speed dials
400	Nonexistent parameters : <destination> <id> does not exist	The destination you are trying to associate with does not exist

Example request

```
POST /1.1/func_keys HTTP/1.1
Host: xivoserver
Accept: application/json
Content-Type: application/json

{
     "type": "speeddial",
     "destination": "user",
     "destination_id": 12
}
```

Example response

```
HTTP/1.1 201 Created
Location: /1.1/func_keys/1
Content-Type: application/json

{
     "id": "1",
     "type": "speeddial",
     "destination": "user",
     "destination_id": 12
     "links": [
         {
             "rel": "func_keys",
             "href: "https://xivoserver/1.1/func_keys/1"
         }
     ]
}
```

Delete a Function Key Destination Most function keys are automatically removed upon the deletion of a desintation resource (See Function Key destinations for speed dials for further details). This action is for deleting function keys that cannot be removed automatically (i.e. custom speed dials and other types of function keys)

> **Warning:** Not implemented yet

Errors

Error code	Error message	Description
400	error while deleting Function Key: <explanation>	See error message for more details
404	Not found	The requested function key was not found or does not exist

Query

```
DELETE /1.1/func_keys/<id>
```

Example request

```
DELETE /1.1/func_keys/1 HTTP/1.1
Host: xivoserver
```

Example response

```
HTTP/1.1 204 No Content
```

Lines The resource `/lines/` only provides read operations. Modifications can only be done on protocol-specific lines (see below).

Generic Lines

Line Representation **Description**

Field	Values	Description
id	int	Read-only
context	string	The name of an internal context
name	string	The name of the line
protocol	string, only value: `sip`	Read-only
provisioning_extension	int	Code used to provision a device
device_slot	int	line's position on the device
device_id	string	ID of the device associated to the line
links	list	The links to the resource

SIP example

```
{
    "id": 1,
    "context": "default",
    "name": "a1b2c4",
    "protocol": "sip",
    "provisioning_extension": 342395,
    "device_slot": 1,
    "device_id": "2b63136208fb117335ce874e65eba2a3",
    "links" : [
        {
            "rel": "lines_sip",
            "href": "https://xivoserver/1.1/lines_sip/1"
        }
    ]
}
```

Custom example

Warning: Not yet implemented

```
{
    "id": 2,
    "context": "default",
    "name": "custom",
    "protocol": "custom",
    "provisioning_extension": 438111,
    "device_slot": 2,
    "links" : [
        {
            "rel": "lines_custom",
            "href": "https://xivoserver/1.1/lines_custom/2"
        }
    ]
}
```

SCCP example

Warning: Not yet implemented

```
{
    "id": 3,
    "context": "default",
    "name": "SCCP/1234",
    "protocol": "sccp",
    "provisioning_extension": 382731,
    "device_slot": 1,
    "links" : [
        {
            "rel": "lines_sccp",
```

```
            "href": "https://xivoserver/1.1/lines_sccp/3"
        }
    ]
}
```

List Lines Query:

```
GET /1.1/lines
```

Example request:

```
GET /1.1/lines HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```


```
{
    "total": 3,
    "items": [
        {
            "id": 1,
            "context": "default",
            "name": "a1b2c4",
            "protocol": "sip",
            "provisioning_extension": 342395,
            "device_slot": 1,
            "device_id": "2b63136208fb117335ce874e65eba2a3",
            "links" : [
                {
                    "rel": "lines_sip",
                    "href": "https://xivoserver/1.1/lines_sip/1"
                }
            ]
        },
        {
            "id": 2,
            "context": "default",
            "name": "custom",
            "protocol": "custom",
            "provisioning_extension": 438111,
            "device_slot": 2,
            "device_id": "4c63136208fb117g35ce874e6eeba25e",
            "links" : [
                {
                    "rel": "lines_custom",
                    "href": "https://xivoserver/1.1/lines_custom/2"
                }
            ]
        },
        {
            "id": 3,
            "context": "default",
            "name": "SCCP/1234",
            "protocol": "sccp",
            "provisioning_extension": 382731,
            "device_slot": 1,
            "device_id": "3s631t620gfb717835ce8a4e6efba85g",
            "links" : [
                {
```

```
                    "rel": "lines_sccp",
                    "href": "https://xivoserver/1.1/lines_sccp/3"
                }
            ]
        }
    ]
}
```

Get Line Query:

```
GET /1.1/lines/<line_id>
```

Example request:

```
GET /1.1/lines/42 HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 42,
    "context": "default",
    "name": "a1b2c4",
    "protocol": "sip",
    "provisioning_extension": 342395,
    "device_slot": 1,
    "device_id": "2b63136208fb117335ce874e65eba2a3",
    "links" : [
        {
            "rel": "lines_sip",
            "href": "https://xivoserver/1.1/lines_sip/42"
        }
    ]
}
```

SIP Lines

SIP Line Representation Description

Field	Value	Description
id	int	Read-only
context	string	
username	string	Read-only
secret	string	Read-only
provisioning_extension	int	Read-only
device_slot	int	Line's position on the device
callerid	string	Read-only
links	list	The link to the resource

List SIP Lines Query:

```
GET /1.1/lines_sip
```

Example request:

```
GET /1.1/lines_sip HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "total": 2,
    "items": [
        {
            "id": 1,
            "context": "default",
            "username": "abcdef",
            "secret": "secret_password",
            "provisioning_extension": 123456,
            "device_slot": 1,
            "callerid": "\"John Doe\" <1002>",
            "links" : [
                {
                    "rel": "lines_sip",
                    "href": "https://xivoserver/1.1/lines_sip/1"
                }
            ]
        },
        {
            "id": 2,
            "context": "default",
            "username": "stuvwx",
            "secret": "super_secret_password",
            "provisioning_extension": 987456,
            "device_slot"; 1,
            "callerid": "\"Mary Lin\" <1003>",
            "links" : [
                {
                    "rel": "lines_sip",
                    "href": "https://xivoserver/1.1/lines_sip/2"
                }
            ]
        }
    ]
}
```

Get SIP Line **Query**:

```
GET /1.1/lines_sip/<id>
```

Example request:

```
GET /1.1/lines_sip/1 HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
     "id": 1,
     "context": "default",
```

```
    "username": "abcdef",
    "secret": "secret_password",
    "provisioning_extension": 123456,
    "device_slot": 1,
    "callerid": "\"John Doe\" <1002>",
    "links": [
        {
            "rel": "lines_sip",
            "href": "https://xivoserver/1.1/lines_sip/1"
        }
    ]
}
```

Create SIP Line The username, secret and provisioning_extension are autogenerated.

Query:

```
POST /1.1/lines_sip
```

Input

Field	Required	Description
context	yes	
device_slot	yes	Line position on device

Errors

Error code	Error message	Description
400	error while creating Line: <explanation>	See explanation for more details
400	Invalid parameters: context <context> does not exist	
400	Invalid parameters: device_slot must be numeric	Use a positive number for the device slot

Example request:

```
POST /1.1/lines_sip HTTP/1.1
Host: xivoserver
Accept: application/json
Content-Type: application/json

{
    "context": "default"
    "device_slot": 1
}
```

Example response:

```
HTTP/1.1 201 Created
Location: /1.1/lines_sip/1
Content-Type: application/json

 {
    "id": 1,
    "context": "default",
    "username": "abcdef",
    "secret": "secret_password",
    "provisioning_extension": 123456,
    "device_slot": 1,
    "callerid": null,
    "links" : [
        {
            "rel": "lines_sip",
            "href": "https://xivoserver/1.1/lines_sip/1"
```

```
        }
    ]
}
```

Update a SIP Line Only fields that need to be updated should be sent. All other fields will remain unmodified during the update.

Query:

```
PUT /1.1/lines_sip/<id>
```

Errors

Same as for creating a SIP line. Please see Create SIP line

Example request:

```
PUT /1.1/lines_sip/67 HTTP/1.1
Host: xivoserver
Content-Type: application/json

{
    "context": "my_context"
}
```

Example response:

```
HTTP/1.1 204 No Content
```

Delete SIP Line A SIP line can not be deleted if it is still associated with a user, an extesion, or a device. Any user, extension, or device attached to the line must be dissociated first. Consult the documentation on *User Line Association*, *Line Extension Association* and *Devices* for further explanations.

Query:

```
DELETE /1.1/lines_sip/<id>
```

Errors

Error code	Error message	Description
400	error while deleting Line: <explanation>	See error message for more details
400	Error while deleting Line: line still has a link	Line is still associated to a user, extension, or device (see explanation above)
404	Line with line_id=X does not exist	The requested line was not found

Example request:

```
DELETE /1.1/lines_sip/1 HTTP/1.1
Host: xivoserver
```

Example response:

```
HTTP/1.1 204 No Content
```

User-Line Association See *User Line Association*

Line-Extension Association See *Line Extension Association*.

Line Extension Association

Association Representation

Description

Field	Value	Description
line_id	int	Line's ID. Read-only.
extension_id	int	Extension's ID.

Get the Extension associated to a Line

Query

```
GET /lines/<line_id>/extension
```

Errors

Error code	Error message	Description
404	Line with id=<line_id> does not exist	
404	Line with id=<line_id> does not have an extension	

Example Request

```
GET /lines/34/extension
Host: xivoserver
Accept: application/json
```

Example Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "line_id": 34,
    "extension_id": 12,
    "links": [
        {
            "rel": "lines_sip",
            "href": "https://xivoserver/1.1/lines_sip/34"
        },
        {
            "rel": "extensions",
            "href": "https://xivoserver/1.1/extensions/12"
        }
    ]
}
```

Get the Line associated to an Extension

Query

```
GET /extensions/<extension_id>/line
```

Errors

Error code	Error message	Description
404	Extension with id=<extension_id> does not exist	
404	Extension with id=<extension_id> does not have a line	

Example Request

```
GET /extensions/48/line
Host: xivoserver
Accept: application/json
```

Example Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "line_id": 34,
    "extension_id": 48,
    "links": [
        {
            "rel": "lines_sip",
            "href": "https://xivoserver/1.1/lines_sip/34"
        },
        {
            "rel": "extensions",
            "href": "https://xivoserver/1.1/extensions/48"
        }
    ]
}
```

Associate an Extension to a Line

Query

```
POST /lines/<line_id>/extension
```

Input

Field	Required	Values	Description
extension_id	yes	int	Must be an existing id

Errors

Error code	Error message	Description
404	Line with id=<line_id> does not exist	
400	Nonexistent parameters: extension_id <extension_id> does not exist	
400	Invalid parameters: line with id <line_id> already has an extension	You must unassociate the current extension before reassociating a new one

Example request

```
POST /1.1/lines/75/extension
Host: xivoserver
Content-Type: application/json

{
    "extension_id": 46
}
```

Example response

```
HTTP/1.1 201
Location: /1.1/lines/75/extension

{
    "line_id": 75,
    "extension_id": 46,
    "links": [
       {
           "rel": "lines_sip",
           "href": "https://xivoserver/1.1/lines_sip/75"
```

```
        },
        {
            "rel": "extensions",
            "href": "https://xivoserver/1.1/extensions/46"
        }
    ]
}
```

Dissociate an Extension from a Line Any devices that are attached to the line must be removed before dissociating an extension from its line. A device can be dissociated be resetting it to autoprov mode. Consult the documentation on *Devices* for further details.

Query

```
DELETE /1.1/lines/<line_id>/extension
```

Errors

Error code	Error message	Description
404	Line with id=<line_id> does not exist	
400	Invalid parameters: A device is still associated to the line	

Example request

```
DELETE /1.1/lines/<line_id>/extension
Host: xivoserver
```

Example response

```
HTTP/1.1 204 No Content
```

Line Extension Associations Connects an extension with a line, allowing the line to be called by dialing a number. A line can be associated with one or more extensions. The context of an extension determines from what source a call can arrive

Currently, this service only supports extensions inside the following context types:

internal Used for calling a line with an internal number (e.g. “1000@default”)

incall Used for calling a line from the outside (e.g. “from-extern” with a DID)

Association Representation

Field	Value	Description
line_id	int	Line’s ID.
extension_id	int	Extension’s ID.

Get the Extension associated to a Line

Query

```
GET /lines/<line_id>/extensions
```

Errors

Error code	Error message	Description
404	Line with id=<line_id> does not exist	

Example Request

```
GET /lines/34/extensions
Host: xivoserver
Accept: application/json
```

Example Response

```
HTTP/1.1 200 OK
Content-Type: application/json
```


```
{
    "total": 2,
    "items":
    [
        {
            "line_id": 34,
            "extension_id": 12,
            "links": [
                {
                    "rel": "lines",
                    "href": "https://xivoserver/1.1/lines/34"
                },
                {
                    "rel": "extensions",
                    "href": "https://xivoserver/1.1/extensions/12"
                }
            ]
        },
        {
            "line_id": 34,
            "extension_id": 13,
            "links": [
                {
                    "rel": "lines",
                    "href": "https://xivoserver/1.1/lines/34"
                },
                {
                    "rel": "extensions",
                    "href": "https://xivoserver/1.1/extensions/13"
                }
            ]
        }
    ]
}
```


Get the Lines associated to an Extension

Query

```
GET /extensions/<extension_id>/lines
```

Errors

Error code	Error message	Description
404	Extension with id=<extension_id> does not exist	

Example Request

```
GET /extensions/48/lines
Host: xivoserver
Accept: application/json
```

Example Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
```

```
    "total": 2,
    "items":
    [
        {
            "line_id": 34,
            "extension_id": 48,
            "links": [
                {
                    "rel": "lines",
                    "href": "https://xivoserver/1.1/lines/34"
                },
                {
                    "rel": "extensions",
                    "href": "https://xivoserver/1.1/extensions/48"
                }
            ]
        },
        {
            "line_id": 35,
            "extension_id": 48,
            "links": [
                {
                    "rel": "lines",
                    "href": "https://xivoserver/1.1/lines/35"
                },
                {
                    "rel": "extensions",
                    "href": "https://xivoserver/1.1/extensions/48"
                }
            ]
        }
    ]
}
```


Associate an Extension to a Line

Note: Because of technical limitations, a line can only have a single 'internal' extension associated (i.e. an extension with a context of type 'internal')

Query

```
POST /lines/<line_id>/extensions
```

Input

Field	Required	Values	Description
extension_id	yes	int	Must be an existing id

Errors

Error code	Error message	Description
404	Line with id=<line_id> does not exist	
400	Invalid parameters: line with id <line_id> already has an extension with a context of type 'internal'	Only one extension with a context of type 'internal' can be associated to a line

Example request

```
POST /1.1/lines/75/extensions
Host: xivoserver
Content-Type: application/json

{
```

```
    "extension_id": 46
}
```

Example response

```
HTTP/1.1 201
Location: /1.1/lines/75/extension


{
    "total": 1,
    "items":
    [
        {
            "line_id": 75,
            "extension_id": 46,
            "links": [
                {
                    "rel": "lines",
                    "href": "https://xivoserver/1.1/lines/75"
                },
                {
                    "rel": "extensions",
                    "href": "https://xivoserver/1.1/extensions/46"
                }
            ]
        }
    ]
}
```

Dissociate an Extension from a Line Any devices that are attached to a line must be removed before dissociating an extension from its line. A device can be dissociated by resetting it to autoprov mode. Consult the documentation on *Devices* for further details.

Query

```
DELETE /1.1/lines/<line_id>/extensions/<extension_id>
```

Errors

Error code	Error message	Description
404	Line with id=<line_id> does not exist	
404	Extension with id=<extension_id> does not exist	
400	Invalid parameters: A device is still associated to the line	

Example request

```
DELETE /1.1/lines/32/extensions/16
Host: xivoserver
```

Example response

```
HTTP/1.1 204 No Content
```

Users

User Representation

Description

Field	Values	Description
id	int	Read-only
firstname	string	User's first name
lastname	string	User's last name
timezone	string	User's timezone
language	string	User's language
description	string	Additional information about the user
caller_id	string	Name that appears on the phone when calling
outgoing_caller_id	string	Caller id to use when calling through a trunk
mobile_phone_number	string	Phone number for the user's mobile device
username	string	username for connecting to the CTI
password	string	password for connecting to the CTI
music_on_hold	string	Name of the MOH category to use for music on hold
preprocess_subroutine	string	Name of the subroutine to execute in asterisk before receiving a call
userfield	string	A custom field which purpose is left to the client. If the user has no userfield, then this field is an empty string.

Example

```
{
    "id": 1,
    "firstname": "John",
    "lastname": "Doe",
    "timezone": "America/Montreal",
    "language": "fr_FR",
    "description": "The most common name in America",
    "caller_id": "Johnny",
    "outgoing_caller_id": "default",
    "mobile_phone_number": "5554151234",
    "username": "john",
    "password": "supersecretpassword",
    "music_on_hold": "waiting",
    "preprocess_subroutine": "ivr",
    "userfield": ""
}
```

List Users The users are listed in ascending order on lastname, then firstname.

Query

```
GET /1.1/users
```

Parameters

> **Warning:** filtering on the line number is not implemented yet

q List only users matching this filter. The filter is done on the firstname, lastname and firstname + lastname and is case insensitive.

Example requests Listing all available users:

```
GET /1.1/users HTTP/1.1
Host: xivoserver
Accept: application/json
```

Searching for a user called "john":

```
GET /1.1/users?q=john HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "total": 2,
    "items":
    [
        {
             "id": 1,
             "firstname": "John",
             "lastname": "Doe",
             "timezone": "",
             "language": "en_US",
             "description": "",
             "caller_id": "\"John Doe\"",
             "outgoing_caller_id": "default",
             "mobile_phone_number": "",
             "username": "",
             "password": "",
             "music_on_hold": "default",
             "preprocess_subroutine": "",
             "userfield": ""
        },
        {
             "id": 2,
             "firstname": "Mary",
             "lastname": "Sue",
             "timezone": "",
             "language": "fr_FR",
             "description": "",
             "caller_id": "\"Mary Sue\"",
             "outgoing_caller_id": "default",
             "mobile_phone_number": "",
             "username": "",
             "password": "",
             "music_on_hold": "default",
             "preprocess_subroutine": "",
             "userfield": ""
        }
    ]
}
```

Get User

```
GET /1.1/users/<id>
```

Parameters

include See List Users.

Example request

```
GET /1.1/users/1 HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
            "id": 1,
            "firstname": "John",
            "lastname": "Doe",
            "timezone": "",
            "language": "en_US",
            "description": "",
            "caller_id": "\"John Doe\"",
            "outgoing_caller_id": "default",
            "mobile_phone_number": "",
            "username": "",
            "password": "",
            "music_on_hold": "default",
            "preprocess_subroutine": "",
            "userfield": ""
}
```

Create a User

Query

```
POST /1.1/users
```

Input

Field	Required	Values
firstname	yes	string
lastname	no	string
timezone	no	string. Must be a valid timezone
language	no	string. Must be a valid language
description	no	string
caller_id	no	string
outgoing_caller_id	no	string: default, anonymous or custom
mobile_phone_number	no	string of digits
username	no	string
password	no	string. Minimum of 4 characters
music_on_hold	no	string. Must be a valid category
preprocess_subroutine	no	string
userfield	no	string

Errors

Error code	Error message	Description
400	error while creating User: <explanation>	See error message for more details

Example request

```
POST /1.1/users HTTP/1.1
Host: xivoserver
Accept: application/json
Content-Type: application/json

{
    "firstname": "John",
    "lastname": "Doe",
    "userfield": ""
}
```

Example response

```
HTTP/1.1 201 Created
Location: /1.1/users/1
Content-Type: application/json

{
    "id": 1,
    "firstname": "John",
    "lastname": "Doe",
    "timezone": "",
    "language": "en_US",
    "description": "",
    "caller_id": "\"John Doe\"",
    "outgoing_caller_id": "default",
    "mobile_phone_number": "",
    "username": "",
    "password": "",
    "music_on_hold": "default",
    "preprocess_subroutine": "",
    "userfield": ""
    "links" : [
        {
            "rel": "users",
            "href": "https://xivoserver/1.1/users/1"
        }
    ]
}
```

Update a User Only the fields that need to be modified can be set.

If the firstname or the lastname is modified, the name of associated voicemail is also updated.

Query

```
PUT /1.1/users/<id>
```

Input Same as for creating a User. Please see Create a User

Errors Same as for creating a User. Please see Create a User

Example request

```
PUT /1.1/users/67 HTTP/1.1
Host: xivoserver
Content-Type: application/json

{
    "firstname": "Jonathan"
}
```

Example response

```
HTTP/1.1 204 No Content
```

Delete User A user can not be deleted if he is associated to a line or a voicemail. Any line or voicemail attached to the user must be dissociated first. Consult the documentation on *User Line Association* and *Voicemail Association* for further details.

The user will also be removed from all queues, groups or other XiVO entities whom he is member.

Query

```
DELETE /1.1/users/<id>
```

Errors

Error code	Error message	Description
400	error while deleting User: <explanation>	See error message for more details
400	Error while deleting User: user still associated to a line	See explanation above
400	Error while deleting User: user still associated to a voicemail	See explanation above
404	User with id=X does not exist	The requested user was not found

Example request

```
DELETE /1.1/users/67 HTTP/1.1
Host: xivoserver
```

Example response

```
HTTP/1.1 204 No Content
```

User-Line Association See *User Line Association*.

Users-Voicemails Association See *Voicemail Association*.

Users-CTI profiles Association See *User CTI configuration*.

User CTI configuration

CTI Configuration Representation

Description

Field	Value	Description
user_id	int	User's ID. Read-only.
cti_profile_id	int	CTI Profile's ID.
enabled	bool	Status of the CTI configuration

Get the CTI Configuration for a User

Query

```
GET /users/<user_id>/cti
```

Errors

Error code	Error message	Description
404	User with id=<user_id> does not exist	

Example Request

```
GET /users/34/cti
Host: xivoserver
Accept: application/json
```

Example Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "user_id": 34,
    "cti_profile_id": 2,
    "links": [
        {
            "rel": "users",
            "href": "https://xivoserver/1.1/users/34"
        },
        {
            "rel": "cti_profiles",
            "href": "https://xivoserver/1.1/cti_profiles/2"
        }
    ]
}
```

Edit the CTI configuration of a user

Query

```
PUT /users/<user_id>/cti
```

Input

Field	Required	Values	Description
cti_profile_id	yes	int	Must be an existing id
enabled	yes	bool	Enable / disable the CTI

Errors

Error code	Error message	Description
404	User with id=<user_id> does not exist	
400	Nonexistent parameters: cti_profile_id <cti_profile_id> does not exist	
400	Error while editing <user_id> : the user must have a username and password to enable the CTI	Add a username and a password to the user

Example request

```
PUT /1.1/users/75/cti
Host: xivoserver
Content-Type: application/json

{
    "cti_profile_id": 3,
    "enabled": true
}
```

Example response

```
HTTP/1.1 204 No Content
```

User Line Association Service for associating a user with a line.

Association Representation

Description

Field	Value	Description
line_id	bool	Line's ID
main_user	boolean	Read-only. True if the user is the first to have been associated to the line.
main_line	boolean	Read-only. To be implemented later. Always true.
links	list	The links to the related resources

Get the Lines associated to a User

Query

```
GET /1.1/users/<user_id>/lines
```

Errors

Error code	Error message	Description
404	User with id=<user_id> does not exist	

Example request

```
GET /1.1/users/20/lines
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
     "total": 1,
     "items": [
         {
             "user_id": 20,
             "line_id": 132,
             "main_user": true,
             "main_line": true,
             "links": [
                 {
                     "rel": "lines",
                     "href": "https://xivoserver/1.1/lines/132"
                 },
                 {
                     "rel": "users",
                     "href": "https://xivoserver/1.1/users/20"
                 }
             ]
         }
     ]
}
```

Associate a Line to a User

Query

```
POST /1.1/users/<user_id>/lines
```

Input

Field	Required	Values	Description
line_id	yes	int	Must be an existing id

Errors

Error code	Error message	Description
400	Nonexistent parameters: user_id <user_id> does not exist	
400	Nonexistent parameters: line_id <line_id> does not exist	
400	Invalid parameters: user is already associated to this line	
400	Invalid parameters: There is an extension associated to this line	

Example request

```
POST /1.1/users/59/lines
Host: xivoserver
Content-Type: application/json

{
    "line_id": 432
}
```

Example response

```
HTTP/1.1 201
Location: /1.1/users/59/lines

{
    "user_id": 59,
    "line_id": 432,
    "main_user": true,
    "main_line": true,
    "links": [
       {
           "rel": "lines",
           "href": "https://xivoserver/1.1/lines/432"
       },
       {
           "rel": "users",
           "href": "https://xivoserver/1.1/users/59"
       }
    ]
}
```

Dissociate a User from a Line Any devices that are attached the line must be removed before dissociating a user from its line. A device can be dissociated be resetting it to autoprov mode. Consult the documentation on *Devices* for further details.

Query

```
DELETE /1.1/users/<user_id>/lines/<line_id>
```

Errors

Error code	Error message	Description
400	User with id=<user_id> is not associated with line id=<line_id>	
400	Invalid parameters: There are secondary users associated to this user_line	
400	Invalid parameters: A device is still associated to the line	

Example request

```
DELETE /1.1/users/59/lines/598
Host: xivoserver
Content-Type: application/json
```

Example response

```
HTTP/1.1 204 No Content
```

Voicemails

Voicemail Representation

Description

Field	Values	Description
id	integer	(Read-only)
name	string	Voicemail name.
number	string	Voicemail number.
context	string	Voicemail's context.
password	string	Numeric password used to access the mailbox
email	string	Email address where messages will be sent.
language	string	Language used for the voicemail menu prompt. See Voicemail languages for a list of available languages.
timezone	string	Timezone used for announcing at what time a message was recorded. See Voicemail timezones for a list of timezones.
max_messages	integer	Maximum number of messages to store.
attach_audio	boolean	Attach an audio file of the recorded message when sending an email.
delete_messages	boolean	Delete messages once they have been listened to.
ask_password	boolean	Ask for password when accessing the voicemail menu.

Example

```
{
    "id": "1",
    "name": "John Doe",
    "number": "1000",
    "context": "default",
    "password": "1234",
    "email": "john.doe@example.com",
    "language": "en_US",
    "timezone": "eu-fr",
    "max_messages": 10,
    "attach_audio: false,
    "delete_messages": false,
    "ask_password": true,
    "links": [
         {
             "rel": "voicemails",
             "href: "https://xivoserver/1.1/voicemails/1"
         }
     ]
 }
```

Voicemail list

Query

```
GET /1.1/voicemails
```

Parameters

order Sort the list using a column (e.g. "number"). Columns allowed: name, number, context, email, language, timezone.

direction 'asc' or 'desc'. Sort list in ascending (asc) or descending (desc) order

limit total number of voicemails to show in the list. Must be a positive integer

skip number of voicemails to skip over before starting the list. Must be a positive integer

search Search voicemails. Only voicemails with a field containing the search term will be listed.

Errors

Error code	Error message	Description
400	Invalid parameters: limit must be a positive number	the 'limit' parameter must be a number
400	Invalid parameters: skip must be a positive number	the 'skip' parameter must be a number
400	Invalid parameters: ordering parameter '<field>' does not exist	you must use one of the fields available in a device when sorting a list
400	Invalid parameters: direction parameter '<direction>' does not exist	use either 'asc' or 'desc' as a direction when sorting a list

Example requests List all available voicemails:

```
GET /1.1/voicemails HTTP/1.1
Host: xivoserver
Accept: application/json
```

List voicemails, sort by descending number:

```
GET /1.1/voicemails?order=number&direction=desc
Host: xivoserver
Accept: application/json
```

List only the first 10 voicemails containing the word "john":

```
GET /1.1/voicemails?search=john&limit=10
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "total": 2,
    "items": [
         {
             "id": "1",
             "name": "John Doe",
             "number": "1000",
             "context": "default",
             "password": null,
             "email": "john.doe@example.com",
             "language": "en_US",
             "timezone": "eu-fr",
             "max_messages": null,
             "attach_audio: false,
             "delete_messages": false,
             "ask_password": false,
```

```
            "links": [
                {
                    "rel": "voicemails",
                    "href: "https://xivoserver/1.1/voicemails/1"
                }
            ]
        },
        {
            "id": "2",
            "name": "Roger Smith",
            "number": "1001",
            "context": "default",
            "password": null,
            "email": null,
            "language": "en_US",
            "timezone": "eu-fr",
            "max_messages": 20,
            "attach_audio: false,
            "delete_messages": false,
            "ask_password": false,
            "links": [
                {
                    "rel": "voicemails",
                    "href: "https://xivoserver/1.1/voicemails/2"
                }
            ]
        }
    ]
}
```


Get Voicemail

Query

```
GET /1.1/voicemails/<id>
```

Example request

```
GET /1.1/voicemails/1 HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json
```


```
{
    "id": "1",
    "name": "John Doe",
    "number": "1000",
    "context": "default",
    "password": null,
    "email": "john.doe@example.com",
    "language": "en_US",
    "timezone": "eu-fr",
    "max_messages": null,
    "attach_audio: false,
    "delete_messages": false,
    "ask_password": false,
    "links": [
```

```
            {
                "rel": "voicemails",
                "href: "https://xivoserver/1.1/voicemails/2"
            }
        ]
}
```

Create a Voicemail

Query

`POST /1.1/voicemails`

Input

Field	Required	Values	Notes
name	yes	string	
number	yes	string	Must be a string of positive numbers
context	yes	string	
password	no	string	Must be a string of positive numbers
email	no	string	
language	no	string	Consult Voicemail Languages for a list of valid languages. The system default will be used if none is specified.
timezone	no	string	Consult Voicemail Timezones for a list of valid timezones. The system default will be used if none is specified.
max_messages	no	integer	Valid values are: 1,10,15,20,25,50,75,100,125,150,175,200,300,400,500,600,700,800,900,1000,2000,3000,40
attach_audio	no	boolean	Default value is *false*
delete_messages	no	boolean	Default value is *false*
ask_password	no	boolean	Default value is *false*

Errors

Error code	Error message	Description
500	Error while creating Voicemail: <explanation>	See explanation for more details.
400	Error while creating Voicemail: number <number> already exists	A voicemail with the same number already exists. Use another number.
400	Invalid parameters: password	Only numeric passwords are supported.
400	Invalid parameters: number <number> must be a sequence of positive numbers	The string must only have positive numbers
400	Invalid parameters: max_messages must be greater than 0	Only positive integers are accepted.
400	Nonexistent parameters: context <context> does not exist	The context used by the voicemail does not exist. You must create the context first.
400	Nonexistent parameters: language <language> does not exist	Consult Voicemail Languages for a list of available languages.
400	Nonexistent parameters: timezone <timezone> does not exist	Consult Voicemail Timezones for a list of available timezones.
400	Missing parameters: <list of missing fields>	

Example request

```
POST /1.1/voicemails HTTP/1.1
Host: xivoserver
Accept: application/json
```

```
Content-Type: application/json

{
     "name": "John Doe",
     "number": "1000",
     "context": "default"
}
```

Example response

```
HTTP/1.1 201 Created
Location: /1.1/voicemails/1
Content-Type: application/json

{
     "id": "1",
     "name": "John Doe",
     "number": "1000",
     "context": "default",
     "password": null,
     "email": null,
     "language": null,
     "timezone": "eu-fr"
     "max_messages": null,
     "attach_audio: false,
     "delete_messages": false,
     "ask_password": false,
     "links": [
         {
             "rel": "voicemails",
             "href: "https://xivoserver/1.1/voicemails/2"
         }
     ]
}
```

Update a Voicemail Only the fields that need to be updated must be sent during an update. A voicemail can only be updated if it isn't associated to a user.

Query

```
PUT /1.1/voicemails/<id>
```

Parameters

id Voicemail's id

Input Same as for creating a voicemail. Please see Create a Voicemail

Errors Same as creating a voicemail (See Create a Voicemail) with the following additions:

Error code	Error message	Description
400	Error while editing Voicemail: cannot edit a voicemail associated to a user	

Example request

```
PUT /1.1/voicemails/1 HTTP/1.1
Host: xivoserver
Content-Type: application/json
```

```
{
    "number": "2000",
    "attach_audio": true
}
```

Example response

```
HTTP/1.1 204 No Content
```

Delete a Voicemail A voicemail can not be deleted if it is still attached to a user. The user must be dissociated first. Consult the documentation on *Voicemail Association* for futher details.

> **Warning:** Any extension that redirects to the voicemail (e.g. an Incoming call) will be disabled after deletion.

Errors

Error code	Error message	Description
400	error while deleting Voicemail <explanation>	See error message for more details
400	error while deleting Voicemail: Cannot delete a voicemail associated to a user	You must unassociate a user from his voicemail before deleting it
404	Voicemail with uniqueid=X does not exist	The requested voicemail was not found or does not exist

Query

```
DELETE /1.1/voicemails/<id>
```

Example request

```
DELETE /1.1/voicemails/1 HTTP/1.1
Host: xivoserver
```

Example response

```
HTTP/1.1 204 No Content
```

Voicemail Languages

> **Warning:** Not yet implemented.

Returns a list of languages that can be used when creating or updating a voicemail.

Query

```
GET /1.1/voicemails/languages
```

Example request

```
GET /1.1/voicemails/languages HTTP/1.1
Host: xivoserver
Content-Type: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
```

```
    "total": 7,
    "items": [
         "de_DE",
         "en_US",
         "es_ES",
         "fr_FR",
         "fr_CA",
         "it_IT",
         "nl_NL"
     ]
 }
```

Voicemail Timezones

Warning: Not yet implemented.

Returns a list of timezones that can be used when creating or updating a voicemail.

Query

```
GET /1.1/voicemails/timezones
```

Example request

```
GET /1.1/voicemails/timezones HTTP/1.1
Host: xivoserver
Content-Type: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "total": 1,
    "items": [
         "eu-fr"
     ]
}
```

Voicemail Association Service for associating a user with a voicemail.

Association Representation

Description

Field	Value	Description
voicemail_id	int	Voicemail's ID
enabled	bool	Enable voicemail for user

Get the Voicemail associated to a User

Query

```
GET /1.1/users/<user_id>/voicemail
```

Errors

Error code	Error message	Description
404	Invalid parameters: user with id=<user_id> does not have a voicemail	
404	User with id=<user_id> does not exist	

Example request

```
GET /1.1/users/20/voicemail
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "voicemail_id": 132,
    "user_id": 20,
    "enabled": true,
    "links": [
       {
           "rel": "voicemails",
           "href": "https://xivoserver/1.1/voicemails/132"
       },
       {
           "rel": "users",
           "href": "https://xivoserver/1.1/users/20"
       }
    ]
}
```

Associate a User to a Voicemail

Query

```
POST /1.1/users/<user_id>/voicemail
```

Input

Field	Required	Values	Description
voicemail_id	yes	int	Must be an existing id
enabled	no	bool	Default value : true

Errors

Error code	Error message	Description
400	Nonexistent parameters: voicemail_id <voicemail_id> does not exist	
400	Invalid parameters: user with id <user_id> does not have any line	A user needs to have a line to associate a voicemail
400	Invalid parameters: user with id <user_id> already has a voicemail	You must unassociate the current voicemail before reassociating a new one

Example request

```
POST /1.1/users/59/voicemail
Host: xivoserver
Content-Type: application/json

{
    "voicemail_id": 432,
    "enabled": false,
}
```

Example response

```
HTTP/1.1 201
Location: /1.1/users/59/voicemail

{
    "voicemail_id": 432,
    "user_id": 59,
    "enabled": false,
    "links": [
       {
           "rel": "voicemails",
           "href": "https://xivoserver/1.1/voicemails/432"
       },
       {
           "rel": "users",
           "href": "https://xivoserver/1.1/users/59"
       }
    ]
}
```

Deassociate a User from a Voicemail

Query

```
DELETE /1.1/users/<user_id>/voicemail
```

Example request

```
DELETE /1.1/users/20/voicemail
Host: xivoserver
```

Example response

```
HTTP/1.1 204 No Content
```

Configuration

Configuration parameters

Parameter	Values	Description
live_reload	bool	

Get live reload status

Query

```
GET /1.1/configuration/live_reload
```

Example requests

```
GET /1.1/configuration/live_reload HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "enabled": true
    "links": [
         {
             "rel": "configuration",
             "href": "https://xivoserver/1.1/configuration/live_reload"
         }
     ]
 }
```

Change live reload status

Query

```
PUT /1.1/configuration/live_reload
```

Example request

```
PUT /1.1/configuration/live_reload HTTP/1.1
Host: xivoserver
Content-Type: application/json

{
   "enabled": false
}
```

Example response

```
HTTP/1.1 204 No Content
```

Migration from 1.0

URL

- Occurences of `1.0` have been replaced for `1.1`
- Trailing slashes have been removed.

  For example, in 1.0, the URL to list users is:

  `/1.0/users/`

  In 1.1, it is:

  `/1.1/users`

1.11.5 Subroutine

What is it ?

The preprocess subroutine allows you to enhance XiVO features through the Asterisk dialplan. Features that can be enhanced are :

- User
- Group

- Queue
- Meetme
- Incoming call
- Outgoing call

There are three possible categories :

- Subroutine for one feature
- Subroutine for global forwarding
- Subroutine for global incoming call to an object

Adding new subroutine

If you want to add a new subroutine, we propose to edit a new configuration file in the directory `/etc/asterisk/extensions_extra.d`. You can also add this file by the web interface.

An example:

```
[myexemple]
exten = s,1,NoOp(This is an example)
same  =   n,Return()
```

Don't forget to finish your subroutine by a Return().

Global subroutine

There is predefined subroutine for this feature, you can find the name and the activation in the `/etc/xivo/asterisk/xivo_globals.conf`. The variables are:

```
; Global Preprocess subroutine
XIVO_PRESUBR_GLOBAL_ENABLE = 1
XIVO_PRESUBR_GLOBAL_USER = xivo-subrgbl-user
XIVO_PRESUBR_GLOBAL_AGENT = xivo-subrgbl-agent
XIVO_PRESUBR_GLOBAL_GROUP = xivo-subrgbl-group
XIVO_PRESUBR_GLOBAL_QUEUE = xivo-subrgbl-queue
XIVO_PRESUBR_GLOBAL_MEETME = xivo-subrgbl-meetme
XIVO_PRESUBR_GLOBAL_DID = xivo-subrgbl-did
XIVO_PRESUBR_GLOBAL_OUTCALL = xivo-subrgbl-outcall
XIVO_PRESUBR_GLOBAL_PAGING = xivo-subrgbl-paging
```

So if you want to add a subroutine for all of your XiVO users you can do this:

```
[xivo-subrgbl-user]
exten = s,1,NoOp(This is an example for all my users)
same  =   n,Return()
```

Forward subroutine

You can also use a global subroutine for call forward.

```
; Preprocess subroutine for forwards
XIVO_PRESUBR_FWD_ENABLE = 1
XIVO_PRESUBR_FWD_USER = xivo-subrfwd-user
XIVO_PRESUBR_FWD_GROUP = xivo-subrfwd-group
XIVO_PRESUBR_FWD_QUEUE = xivo-subrfwd-queue
XIVO_PRESUBR_FWD_MEETME = xivo-subrfwd-meetme
XIVO_PRESUBR_FWD_VOICEMAIL = xivo-subrfwd-voicemail
XIVO_PRESUBR_FWD_SCHEDULE = xivo-subrfwd-schedule
```

```
XIVO_PRESUBR_FWD_VOICEMENU = xivo-subrfwd-voicemenu
XIVO_PRESUBR_FWD_SOUND = xivo-subrfwd-sound
XIVO_PRESUBR_FWD_CUSTOM = xivo-subrfwd-custom
XIVO_PRESUBR_FWD_EXTENSION = xivo-subrfwd-extension
```

Dialplan variables

Some of the XiVO variables can be used in subroutines.

```
XIVO_CALLORIGIN ; intern for internal calls, extern for external calls
```

1.12 Developers

General information:

1.12.1 Contributing to the Documentation

XiVO documentation is generated with Sphinx. The source code is available on GitHub at https://github.com/xivo-pbx/xivo-doc

Provided you already have Python installed on your system. You need first to install Sphinx : `easy_install -U Sphinx` [11].

Quick Reference

- http://docutils.sourceforge.net/docs/user/rst/cheatsheet.txt
- http://docutils.sourceforge.net/docs/user/rst/quickref.html
- http://openalea.gforge.inria.fr/doc/openalea/doc/_build/html/source/sphinx/rest_syntax.html

Documentation guideline

Here's the guideline/conventions to follow for the XiVO documentation.

Language

The documentation must be written in english, and only in english.

Sections

The top section of each file must be capitalized using the following rule: capitalization of all words, except for articles, prepositions, conjunctions, and forms of to be.

Correct:

```
The Vitamins are in My Fresh California Raisins
```

Incorrect:

```
The Vitamins Are In My Fresh California Raisins
```

Use the following punctuation characters:

- `*` with overline, for "file title"

[11] `easy_install` can be found in the debian package python-setuptools : `sudo apt-get install python-setuptools`

- =, for sections
- -, for subsections
- ^, for subsubsections

Punctuation characters should be exactly as long as the section text.

Correct:

```
Section1
========
```

Incorrect:

```
Section2
==========
```

There should be 2 empty lines between sections, except when an empty section is followed by another section.

Correct:

```
Section1
========

Foo.


Section2
========

Bar.
```

Correct:

```
Section1
========

Foo.


.. _target:

Section2
========

Bar.
```

Correct:

```
Section1
========

Subsection1
-----------

Foo.
```

Incorrect:

```
Section1
========

Foo.

Section2
========
```

```
Bar.
```

Lists

Bullet lists:

```
* First item
* Second item
```

Autonumbered lists:

```
#. First item
#. Second item
```

Literal blocks

Use `::` on the same line as the line containing text when possible.

The literal blocks must be indented with three spaces.

Correct:

```
Bla bla bla::

   apt-get update
```

Incorrect:

```
Bla bla bla:

::

   apt-get update
```

Inline markup

Use the following roles when applicable:

- `:file:` for file, i.e.:

  ```
  The :file:`/dev/null` file.
  ```

- `:menuselection:` for the web interface menu:

  ```
  The :menuselection:`Configuration --> Management --> Certificates` page.
  ```

- `:guilabel:` for designating a specific GUI element:

  ```
  The :guilabel:`Action` column.
  ```

Others

- There must be no warning nor error messages when building the documentation with `make html`.
- There should be one and only one newline character at the end of each file
- There should be no trailing whitespace at the end of lines
- Paragraphs must be wrapped and lines should be at most 100 characters long

1.12.2 Debugging Asterisk

To debug asterisk crashes or freezes, you need the following packages on your xivo:

```
apt-get install gdb asterisk-dbg xivo-libsccp-dbg
```

Warning: When installing theses packages you should take care that it doesn't drag a new version of asterisk since it would restart your asterisk

Debugging Asterisk Crash

When asterisk crashes, it usually leaves a core file in `/var/spool/asterisk/`.

You can create a backtrace from a core file named `core_file` with:

```
gdb -batch -ex "bt full" -ex "thread apply all bt" asterisk core_file > bt-threads.txt
```

Debugging Asterisk Freeze

You can create a backtrace of a running asterisk process with:

```
gdb -batch -ex "thread apply all bt" asterisk $(pidof asterisk) > bt-threads.txt
```

If your version of asterisk has been compiled with the DEBUG_THREADS flag, you can get more information about locks with:

```
asterisk -rx "core show locks" > core-show-locks.txt
```

Note: Debugging freeze without this information is usually a lot more difficult.

Optionally, other information that can be interesting:

- the output of `asterisk -rx 'core show channels'`
- the verbose log of asterisk just before the freeze

Recompiling Asterisk

It's relatively straightforward to recompile the asterisk version of your xivo with the DEBUG_THREADS and DONT_OPTIMIZE flag, which make debugging an asterisk problem easier.

The steps are:

1. Uncomment the `deb-src` line for the xivo sources:

   ```
   sed -i 's/^#deb-src/deb-src/' /etc/apt/sources.list.d/xivo*
   ```

2. Fetch the asterisk source package:

   ```
   mkdir -p ~/ast-rebuild
   cd ~/ast-rebuild
   apt-get update
   apt-get source asterisk
   ```

3. Install the build dependencies:

   ```
   apt-get install build-essential
   apt-get build-dep asterisk
   ```

4. Enable the DEBUG_THREADS and DONT_OPTIMIZE flag:

```
cd <asterisk-source-folder>
vim debian/rules
```

5. Update the changelog by appending `+debug1` in the package version:

   ```
   vim debian/changelog
   ```

6. Rebuild the asterisk binary packages:

   ```
   dpkg-buildpackage -us -uc
   ```

This will create a couple of .deb files in the parent directory, which you can install via dpkg.

External links

- https://wiki.asterisk.org/wiki/display/AST/Debugging
- http://blog.xivo.fr/index.php?post/2012/10/24/Visualizing-asterisk-deadlocks

1.12.3 Debugging Daemons

Here's how to run the various daemons present in XiVO in foreground and debug mode.

Note that it's usually a good idea to stop monit before running a daemon in foreground.

agentd

```
xivo-agentd -f -v
```

- -f for foreground
- -v for verbose

Log file: `/var/log/xivo-agentd.log`

```
2013-10-29 11:03:55,799 [25830] (INFO): Starting xivo-agentd
2013-10-29 11:03:58,632 [25830] (INFO): Executing statuses command
```

agid

```
xivo-agid -f -d
```

- -f for foreground
- -d for debug

Log file: `/var/log/daemon.log`. Lines start with `xivo-agid`.

```
Oct 29 11:03:53 hostname xivo-agid[25724]: xivo-agid starting...
Oct 29 11:03:54 hostname xivo-agid[25724]: executing update command 'update-config'
Oct 29 11:03:54 hostname xivo-agid[25724]: executing update command 'update-phonebook'
```

amid

```
xivo-amid -f -v
```

- -f for foreground
- -v for verbose

Log file: /var/log/xivo-amid.log

```
2014-01-15 10:36:42,372 [5252] (INFO): Starting xivo-amid
2014-01-15 10:36:42,372 [5252] (INFO): Connecting socket
2014-01-15 10:36:42,372 [5252] (INFO): Connecting AMI client to localhost:5038
```

call-logd

```
xivo-call-logd -f -v
```

- -f for foreground
- -v for verbose

Log file: /var/log/xivo-call-logd.log

```
2014-02-12 14:58:05,051 [14650] (INFO): Starting xivo-call-logd
2014-02-12 14:58:05,178 [14650] (INFO): Running...
```

confgend

```
twistd -no --python=/usr/bin/xivo-confgend
```

No debug mode in confgend.

Log file: /var/log/xivo-confgend.log

```
2013-10-29 11:03:50-0400 [-] Starting factory <xivo_confgen.confgen.ConfgendFactory instance at 0:
2013-10-29 11:03:55-0400 [Confgen,0,127.0.0.1] serving asterisk/features.conf
2013-10-29 11:03:55-0400 [Confgen,1,127.0.0.1] serving asterisk/musiconhold.conf
```

ctid

```
xivo-ctid -d
```

- -d for foreground and debug

Log file: /var/log/xivo-ctid.log

```
2013-10-29 11:03:58,789 xivo-ctid[25914] (INFO) (main): CTI Fully Booted in 0.660311 seconds
2013-10-29 11:03:58,789 xivo-ctid[25914] (INFO) (interface_ami): Asterisk Call Manager/1.3
2013-10-29 11:03:58,827 xivo-ctid[25914] (INFO) (AMI logger): Event received:Privilege=>system,al
```

dxtora

```
dxtora -f
```

- -f for foreground

Log file: /var/log/daemon.log. Lines start with xivo-dxtora.

```
Oct 28 09:24:48 hostname xivo-dxtora[1399]: Received signal, exiting.
Oct 28 09:24:58 hostname xivo-dxtora[8562]: Pulling DHCP info from unix socket
```

provd

```
twistd -no -r epoll xivo-provd -s -v
```

- -s for logging to stderr

- -v for verbose

Log file: `/var/log/daemon.log`. Lines start with `xivo-provd`.

```
Oct 29 06:24:05 hostname xivo-provd[8596]: TFTP read request from ('192.168.1.1', 53014)
Oct 29 06:24:05 hostname xivo-provd[8596]: Processing TFTP request: i18n/france/7960-tones.xml
Oct 29 06:24:05 hostname xivo-provd[8596]: <14> Extracted device info: {u'ip': u'192.168.1.1'}
Oct 29 06:24:05 hostname xivo-provd[8596]: <14> Retrieved device id: caddf5dcfcc34e088687a6589b63
Oct 29 06:24:05 hostname xivo-provd[8596]: <14> Routing request to plugin xivo-cisco-sccp-9.0.3
```

restapi

```
xivo-restapid -f -d
```

- -f for foreground
- -d for debug messages

Log file: `/var/log/xivo-restapid.log`

```
2013-10-28 10:02:00,352 xivo-restapid[8905] (INFO) (xivo_restapi.flask_http_server): POST http://
2013-10-28 10:04:35,815 xivo-restapid[8905] (INFO) (xivo_restapi.flask_http_server): GET http://1
```

sysconfd

```
xivo-sysconfd -l debug -f
```

- -l debug for debug level logging
- -f for foreground

Log file: `/var/log/daemon.log`. Lines start with `xivo-sysconfd`.

```
Oct 29 11:03:45 hostname xivo-sysconfd[24522]: locking PID
Oct 29 11:03:45 hostname xivo-sysconfd[24522]: pidfile ok
Oct 29 11:03:45 hostname xivo-sysconfd[24522]: will now serve
Oct 29 11:04:33 hostname xivo-sysconfd[24522]: 'GET /status-check HTTP/1.1' 200 17
```

1.12.4 XiVO Guidelines

Inter-process communication

Our current goal is to use only two means of communication between XiVO processes:

- a REST API over HTTP for synchronous commands
- a software bus (RabbitMQ) for asynchronous events

Each component should have its own REST API and its own events and can communicate with every other component from across a network only via those means.

Service API

The current xivo-dao Git repository contains the basis of the future services Python API. The API is split between different resources available in XiVO, such as users, groups, schedules... For each resource, there are different modules :

- service: the public module, providing possible actions. It contains only business logic and no technical logic. There must be no file name, no SQL queries and no URLs in this module.

- dao: the private Data Access Object. It knows where to get data and how to update it, such as SQL queries, file names, URLs, but has no business logic.
- model: the public class used to represent the resource. It must be self-contained and have almost no methods, except for computed fields based on other fields in the same object.
- notifier: private, it knows to whom and in which format events must be sent.
- validator: private, it checks input parameters from the service module.

1.12.5 Profiling Python Programs

Profiling CPU/Time Usage

Here's an example on how to profile xivo-ctid for CPU/time usage:

1. Add the debian non-free repository to `/etc/apt/sources.list`.
2. Install the `python-profiler` package:

   ```
   apt-get update
   apt-get install python-profiler
   ```

3. Stop the monit daemon:

   ```
   /etc/init.d/monit stop
   ```

4. Stop the process you want to profile, i.e. xivo-ctid:

   ```
   /etc/init.d/xivo-ctid stop
   ```

5. Start the service in foreground mode running with the profiler:

   ```
   python -m cProfile -o test.profile /usr/bin/xivo-ctid -d
   ```

   This will create a file named `test.profile` when the process terminates.

   The *Debugging Daemons* section documents how to launch the various XiVO services in foreground/debug mode.

6. Examine the result of the profiling:

   ```
   $ python -m pstats test.profile
   Welcome to the profile statistics browser.
   % sort time
   % stats 15
   ...
   % sort cumulative
   % stats 15
   ```

Measuring Code Coverage

Here's an example on how to measure the code coverage of xivo-ctid.

This can be useful when you suspect a piece of code to be unused and you want to have additional information about it.

1. Install the following packages:

   ```
   apt-get install python-pip build-essential python-dev
   ```

2. Install coverage via pip:

   ```
   pip install coverage
   ```

3. Run the program in foreground mode with `coverage run`:

   ```
   service monit stop
   service xivo-ctid stop
   coverage erase
   coverage run /usr/bin/xivo-ctid -d
   ```

   The *Debugging Daemons* section documents how to launch the various XiVO service in foreground/debug mode.

4. After the process terminates, use `coverage html` to generate an HTML coverage report:

   ```
   coverage html --include='*xivo_cti*'
   ```

   This will generate an `htlmcov` directory in the current directory.

5. Browse the coverage report.

   Either copy the directory onto your computer and open it with a web browser, or start a web server on the XiVO:

   ```
   cd htmlcov
   python -m SimpleHTTPServer
   ```

   Then open the page from your computer (i.e. not on the xivo):

   ```
   firefox http://<xivo-hostname>:8000
   ```

External Links

- Official python documentation
- PyMOTW
- coverage.py

1.12.6 Translating XiVO

Supported languages are French and English. This means that they are always maintained by Avencall. Other languages are provided by the community.

We are now using a centralized tool for the XiVO translation. It's Transifex and the web access is :

- https://www.transifex.net/projects/p/xivo

For the XiVO Client, we have a reference file pushed regularly to Transifex. This file is always by default in english and translated in Transifex.

New strings to be translated are uploaded to Transifex every release. Translated string are downloaded from Transifex every release.

XiVO Prompts

We have different studio for each languages and prompts. The information for those languages are here :

- French : Super Sonic productions (supersonicprod@wanadoo.fr)
- English : Asterisk voice (allison@theasteriskvoice.com)
- German : ATS studio
- Italian : ATS studio

If you want to add a new prompt you need to edit the xivo-prompts-orig.csv in our git xivo-sounds.

Asterisk Prompts

If you want to add a new prompt you need to edit the asterisk-prompts-orig.csv in our git xivo-sounds.

XiVO Client

All translations are in Transifex.

Web Interface

Translations are currently available in French and English. Nothing is currently being done to translate the Web Interface in other languages.

1.12.7 Style Guide

Syntax

License

Python files start with a UTF8 encoding comment and the GPLv3 license. A blank line should separate the license from the imports

Example:

```
# -*- coding: utf-8 -*-

# Copyright (C) 2013 Avencall
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>

import argparse
```

Spacing

- Lines should not go further than 80 to 100 characters.
- In python, indentation blocks use 4 spaces
- In PHP, indentation blocks use tabs
- Imports should be ordered alphabetically
- Separate module imports and `from` imports with a blank line

Example:

```
import argparse
import datetime
import os
import re
import shutil
import tempfile

from StringIO import StringIO
from urllib import urlencode
```

PEP8

When possible, use pep8 to validate your code. Generally, the following errors are ignored :

- E501 (max 80 chars per line)

Example:

```
pep8 --ignore=E501 xivo_cti
```

When possible, avoid using backslashes to separate lines.

Bad Example:

```
user = session.query(User).filter(User.firstname == firstname)\
                          .filter(User.lastname == lastname)\
                          .filter(User.number == number)\
                          .all()
```

Good Example:

```
user = (session.query(User).filter(User.firstname == firstname)
                           .filter(User.lastname == lastname)
                           .filter(User.number == number)
                           .all())
```

Strings

Avoid using the + operator for concatenating strings. Use string interpolation instead.

Bad Example:

```
phone_interface = "SIP" + "/" + username + "-" + password
```

Good Example:

```
phone_interface = "SIP/%s-%s" % (username, password)
```

Comments

Redundant comments should be avoided. Instead, effort should be put on making the code clearer.

Bad Example:

```
#Add the meeting to the calendar only if it was created on a week day
#(monday to friday)
if meeting.day > 0 and meeting.day < 7:
    calendar.add(meeting)
```

Good Example:

```
def created_on_week_day(meeting):
    return meeting.day > 0 and meeting.day < 7

if created_on_week_day(meeting):
    calendar.add(meeting)
```

Conditions

Avoid using parenthesis around if statements, unless the statement expands on multiple lines or you need to nest your conditions.

Bad Examples:

```
if(x == 3):
    print "condition is true"

if(x == 3 and y == 4):
    print "condition is true"
```

Good Examples:

```
if x == 3:
    print "condition is true"

if x == 3 and y == 4:
    print "condition is true"

if (extremely_long_variable == 3
    and another_long_variable == 4
    and yet_another_variable == 5):

    print "condition is true"

if (2 + 3 + 4) - (1 + 1 + 1) == 6:
    print "condition is true"
```

Consider refactoring your statement into a function if it becomes too long, or the meaning isn't clear.

Bad Example:

```
if price * tax - bonus / reduction + fee < money:
    product.pay(money)
```

Good Example:

```
def calculate_price(price, tax, bonus, reduction, fee):
    return price * tax - bonus / reduction + fee

final_price = calculate_price(price, tax, bonus, reduction, fee)

if final_price < money:
    product.pay(money)
```

Naming

- Class names are in `CamelCase`
- File names are in `lower_underscore_case`

Conventions for functions prefixed by *find*:

- Return None when nothing is found

- Return an object when a single entity is found
- Return the first element when multiple entities are found

Example:

```
def find_by_username(username):
    users = [user1, user2, user3]
    user_search = [user for user in users if user.username == username]

    if len(user_search) == 0:
        return None

    return user_search[0]
```

Conventions for functions prefixed by *get*:

- Raise an Exception when nothing is found
- Return an object when a single entity is found
- Return the first element when multiple entities are found

Example:

```
def get_user(userid):
    users = [user1, user2, user3]
    user_search = [user for user in users if user.userid == userid]

    if len(user_search) == 0:
        raise UserNotFoundError(userid)

    return user_search[0]
```

Conventions for functions prefixed by *find_all*:

- Return an empty list when nothing is found
- Return a list of objects when multiple entites are found

Example:

```
def find_all_users_by_username(username):
    users = [user1, user2, user3]
    user_search = [user for user in users if user.username == username]

    return user_search
```

Magic numbers

Magic numbers should be avoided. Arbitrary values should be assigned to variables with a clear name

Bad example:

```
class TestRanking(unittest.TestCase):

    def test_ranking(self):
        rank = Rank(1, 2, 3)

        self.assertEquals(rank.position, 1)
        self.assertEquals(rank.grade, 2)
        self.assertEquals(rank.session, 3)
```

Good example:

```
class TestRanking(unittest.TestCase):

    def test_ranking(self):
        position = 1
        grade = 2
        session = 3

        rank = Rank(position, grade, session)

        self.assertEquals(rank.position, position)
        self.assertEquals(rank.grade, grade)
        self.assertEquals(rank.session, session)
```

Tests

Tests for a package are placed in their own folder named "tests" inside the package.

Example:

```
package1/
__init__.py
mod1.py
tests/
    __init__.py
    test_mod1.py
package2/
__init__.py
mod9.py
tests/
    __init__.py
    test_mod9.py
```

Unit tests should be short, clear and concise in order to make the test easy to understand. A unit test is separated into 3 sections :

- Preconditions / Preparations
- Thing to test
- Assertions

Sections are separated by a blank line. Sections that become too big should be split into smaller functions.

Example:

```
class UserTestCase(unittest.TestCase):

    def test_fullname(self):
        user = User(firstname='Bob', lastname='Marley')
        expected = 'Bob Marley'

        fullname = user.fullname()

        self.assertEquals(expected, fullname)

    def _prepare_expected_user(self, firstname, lastname, number):
        user = User()
        user.firstname = firstname
        user.lastname = lastname
        user.number = number

        return user

    def _assert_users_are_equal(expected_user, actual_user):
```

```
        self.assertEquals(expected_user.firstname, actual_user.firstname)
        self.assertEquals(expected_user.lastname, actual_user.lastname)
        self.assertEquals(expected_user.number, actual_user.number)

    def test_create_user(self):
        expected = self._prepare_expected_user('Bob', 'Marley', '4185551234')

        user = create_user('Bob', 'Marley', '4185551234')

        self._assert_users_are_equal(expected, user)
```

Exceptions

Exceptions should not be used for flow control. Raise exceptions only for edge cases, or when something that isn't usually expected happens.

Bad Example:

```
def is_user_available(user):
    if user.available():
        return True
    else:
        raise Exception("User isn't available")

try:
    is_user_available(user)
except Exception:
    disable_user(user)
```

Good Example:

```
def is_user_available(user):
    if user.available():
        return True
    else:
        return False


if not is_user_available(user):
    disable_user(user)
```

Avoid throwing `Exception`. Use one of Python's built-in Exceptions, or create your own custom Exception. A list of exceptions is available on the Python documentation website.

Bad Example:

```
def get_user(userid):
    user = session.query(User).get(userid)

    if not user:
        raise Exception("User not found")
```

Good Example:

```
class UserNotFoundError(LookupError):

    def __init__(self, userid):
        message = "user with id %s not found" % userid
        LookupError.__init__(self, message)

def get_user(userid):
    user = session.query(User).get(userid)
```

```
    if not user:
        raise UserNotFoundError(userid)
```

Never use `except:` without specifying any exception type. The reason is that it will also catch important exceptions, such as `KeyboardInterrupt` and `OutOfMemory` exceptions, making your program unstoppable or continuously failing, instead of stopping when wanted.

Bad Example:

```
try:
    get_user(user_id)
except:
    logger.exception("There was an error")
```

Good Example:

```
try:
    get_user(user_id)
except UserNotFoundError as e:
    logger.error(e.message)
    raise
```

Component specific information:

1.12.8 CTI Server

This section describes the informations and tools for CTI Server.

CTI Proxy

Here's how to run the various CTI client-server development/debugging tools. These tools can be found on GitHub, in the XiVO project.

You can get the scripts by using Git:

```
$ git clone https://github.com/xivo-pbx/xivo-tools.git
```

General Information

Both the ctispy, ctisave and ctistat tools work in a similar way. They both are proxies that need to be inserted between the CTI client and the CTI server message flow.

To do this, you first start the given tool on your development machine, giving it the CTI server hostname as the first argument. You then configure your CTI client to connect to the tool on port 50030 (notice the trailing 0). The tool should then accept the connection from the client, and once this is done, will make a connection to the server, thereby being able to process all the information sent between the client and the server.

In the following examples, we suppose that the CTI server is located on the host named xivo-new.

Tools

ctispy `ctispy` can be used to see the message flow between the client and the server in "real-time".

The simplest invocation is:

```
$ cti-proxy/ctispy xivo-new
```

You can pretty-print the messages if you want by using the `--pretty-print` option:

```
$ cti-proxy/ctispy xivo-new --pretty-print
```

By default, each message is displayed separately even though more than one message can be in a single TCP packet. You can also use the `--raw` option if you want to see the raw traffic between the client and the server:

```
$ cti-proxy/ctispy xivo-new --raw
```

Note that when using the `--raw` option, some other option doesn't work because the messages are not decoded/analyzed.

If you want to remove some fields from the messages, you can use the `--strip` option:

```
$ cti-proxy/ctispy xivo-new --strip timenow --strip commandid --strip replyid
```

If you want to see only messages matching a certain key and value, use the `--include` option:

```
$ cti-proxy/ctispy xivo-new --include class=getlist
```

Finally, you can ignore all the messages from the client or the server by using the `--no-client` or `--no-server` option respectively.

By default, ctispy will exit after the connection with the client is closed. You can bypass this behavior with the `--loop` option, that will make the CTI proxy continue, whether the client is connected or not.

ctisave `ctisave` save the messages from the client and the server in two separate files. This is useful to do more careful post-analysis.

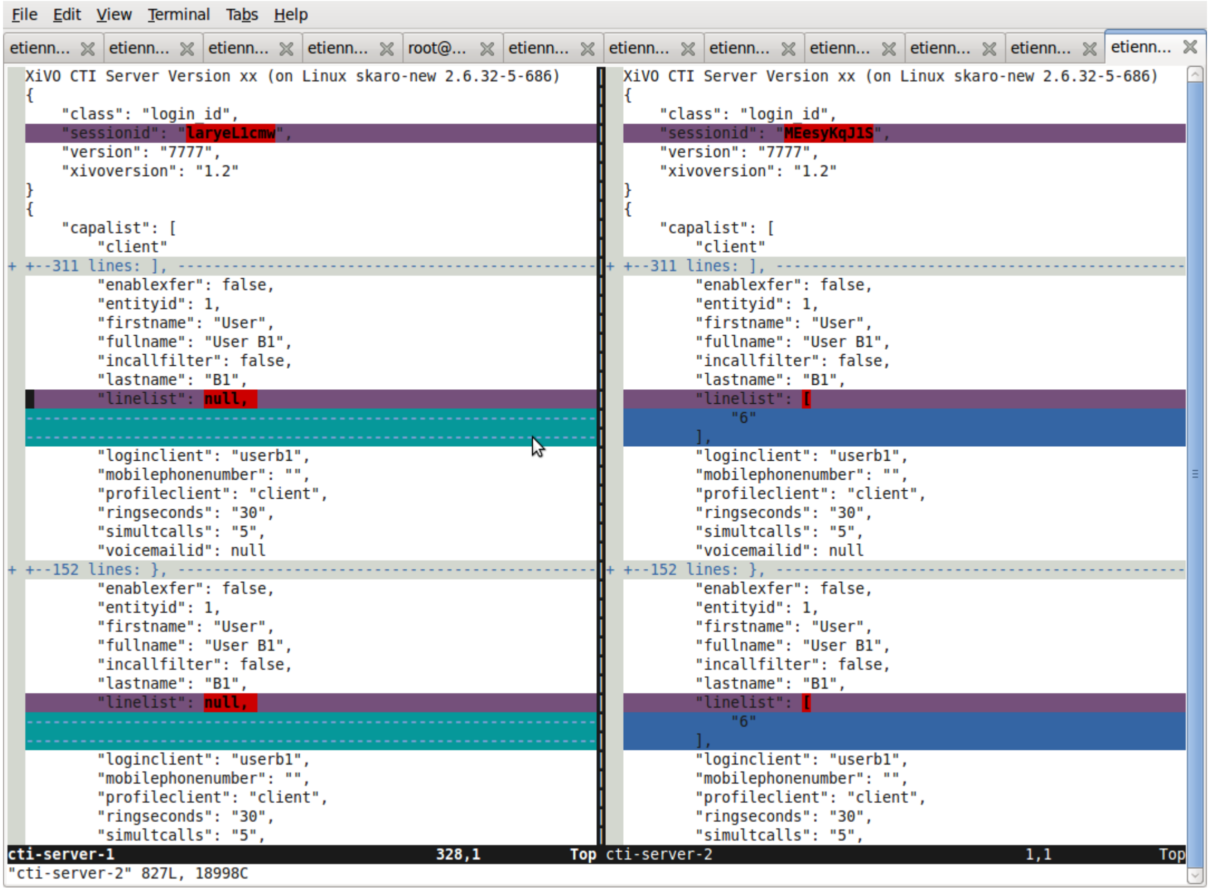
The simplest invocation is:

```
$ cti-proxy/ctisave xivo-new /tmp/cti-client /tmp/cti-server
```

To do comparison, it's often useful to strip some fields:

```
$ cti-proxy/ctisave xivo-new /tmp/cti-client /tmp/cti-server --strip timenow
--strip commandid --strip replyid
```

One useful thing to do with files generated from different ctisave invocation is to compare them with a tool like vimdiff, for example:



ctistat `ctistat` display various statistic about a CTI "session" when it ends.

The simplest invocation is:

```
$ cti-proxy/ctistat xivo-new
```

CTI Protocol

Protocol Changelog

> **Warning:** The CTI server protocol is subject to change without any prior warning. If you are using this protocol in your own tools please be sure to check that the protocol did not change before upgrading XiVO

14.06

- the `dial_success` message was added

14.05

- the `unhold_switchboard` command was renamed `resume_switchboard`.

13.22

- the `actionfiche` message was renamed `call_form_result`.

13.17

- for messages of class `login_capas` from server to client: the key `presence` has been removed.

13.14

- for messages of class `getlist`, list `agents` and function `updatestatus`: the key `availability` in the `status` object/dictionary has changed values:
  - deleted values: `on_call_non_acd_incoming` and `on_call_non_acd_outgoing`
  - added values: * `on_call_non_acd_incoming_internal` * `on_call_non_acd_incoming_external` * `on_call_non_acd_outgoing_internal` * `on_call_non_acd_outgoing_external`

13.12

- for messages of class `getlist`, list `agents` and function `updatestatus`: the key `availability` in the `status` object/dictionary has changed values:
  - deleted value: `on_call_non_acd`
  - added values: `on_call_non_acd_incoming` and `on_call_non_acd_outgoing`

13.10

- for messages of class `getlist` and function `updateconfig`, the `config` object/dictionary does not have a `rules_order` key anymore.

Commands

Objects have the format: "<type>:<xivoid>/<typeid>"

- <type> can take any of the following values: user, agent, queue, phone, group, meetme, ...
- <xivoid> indicates on which server the object is defined
- <typeid> is the object id, type dependant

e.g. user:xivo-test/5 I'm looking for the user that has the ID 5 on the xivo-test server.

Here is a non exaustive list of types:

- exten
- user
- vm_consult
- voicemail

Class list

LOGINCOMMANDS Once the network is connected at the socket level, the login process requires three steps. If one of these steps is omitted, the connection is reseted by the cti server.

- login_id, the username is sent as a login to the cti server, cti server answers by giving a sessionid
- login_pass, the password combined with the sessionid is sent to the cti server, cti server answers by giving a capaid
- login_capas, the capaid is returned to the server with the phone state, cti server answers with a list of info relevevant to the user

```
{
"commandid": <commandid>,
"class": "login_id",
}
```

- class: defined what class of command use.
- commandid : a unique integer number.

login_id `Client -> Server`

```
{
"class": "login_id",
"commandid": 1092130023,
"company": "default",
"ident": "X11-LE-24079",
"lastlogout-datetime": "2013-02-19T11:13:36",
"lastlogout-stopper": "disconnect",
"userlogin": <userlogin>,
"version": "9999",
"xivoversion": "1.2"
}
```

`Server -> Client`

```
{
    "class": "login_id",
    "sessionid": "21UaGDfst7",
    "timenow": 1361268824.64,
    "xivoversion": "1.2"
}
```

Note: sessionid is used to calculate the hashed password in next step

login_pass `Client -> Server`

```
{
"hashedpassword": "e5229ef45824333e0f8bbeed20dccfa2ddcb1c80",
"class": "login_pass",
"commandid": <commandid>
}
```

Note: hashed_password = sha1(self.sessionid + ':' + password).hexdigest()

`Server -> Client`

```
{
    "capalist": [
        2
    ],
    "class": "login_pass",
    "replyid": 1646064863,
    "timenow": 1361268824.68
}
```

If no CTI profile is defined on XiVO for this user, the following message will be sent:

```
{
    "error_string": "capaid_undefined",
    "class": "login_pass",
    "replyid": 1646064863,
```

```
    "timenow": 1361268824.68
}
```

Note: the first element of the capalist is used in the next step login_capas

login_capas `Client -> Server`

```
{
"loginkind": "user",
"capaid": 3,
"lastconnwins": False,
"commandid": <commandid>,
"state": "available",
"class": "login_capas"
}
```

loginkind can be 'user' or 'agent', if 'agent', the property 'agentphonenumber' can be added.

`Server -> Client`

First message, describes all the capabilities of the client, configured at the server level

- presence : actual presence of the user
- userid : the user id, can be used as a reference
- **capas**
  - **userstatus** [a list of available statuses]
    - status name
    - color
    - selectionnable status from this status
    - default action to be done when this status is selected
    - long name
  - services : list of availble services
  - phonestatus : list of available phonestatuses with default colors and descriptive names
  - capaxlets : List of xlets configured for this profile
  - appliname

```
{
   "class": "login_capas"
   "presence": "available",
   "userid": "3",
   "ipbxid": "xivo",
   "timenow": 1361440830.99,
   "replyid": 3,
   "capas": {
            "regcommands": {},
            "preferences": false,
            "userstatus": {
                        "available": { "color": "#08FD20",
                                       "allowed": ["available", "away", "outtolunch", "donotdistu
                                       "actions": {"enabledndd": "false"}, "longname": "Disponible
                                     },
                        "berightback": {   "color": "#FFB545",
                                          "allowed": ["available", "away", "outtolunch", "donotdi
                                          "actions": {"enablednd": "false"}, "longname": "Bient\u
                                       },
```

```
                        "disconnected": { "color": "#202020",
                                          "actions": {"agentlogoff": ""}, "longname": "D\u00e9conn
                                        },
                        /* a list of other status depends on the cti server configuration */
              },
              "services": ["fwdrna", "fwdbusy", "fwdunc", "enablednd"],
              "phonestatus": {
                                "16": {"color": "#F7FF05", "longname": "En Attente"},
                                "1":  {"color": "#FF032D", "longname": "En ligne OU appelle"},
                                "0":  {"color": "#0DFF25", "longname": "Disponible"},
                                "2":  {"color": "#FF0008", "longname": "Occup\u00e9"},
                                "-1": {"color": "#000000", "longname": "D\u00e9sactiv\u00e9"},
                                "4":  {"color": "#FFFFFF", "longname": "Indisponible"},
                                "-2": {"color": "#030303", "longname": "Inexistant"},
                                "9":  {"color": "#FF0526", "longname": "(En Ligne OU Appelle) ET Sonne
                                "8":  {"color": "#1B0AFF", "longname": "Sonne"}
                            },
              "ipbxcommands": {}
          },
    "capaxlets": [["identity", "grid"], ["search", "tab"], ["customerinfo", "tab", "1"], ["fax", "t
    "appliname": "Client",
}
```

Second message describes the current user configuration

```
{
    "function": "updateconfig",
    "listname": "users",
    "tipbxid": "xivo",
    "timenow": 1361440830.99,
    "tid": "3",
    "config": {"enablednd": false},
    "class": "getlist"
}
```

Third message describes the current user status

```
{
    "function": "updatestatus",
    "listname": "users",
    "status": {"availstate": "available"},
    "tipbxid": "xivo",
    "tid": "3",
    "class": "getlist",
    "timenow": 1361440830.99
}
```

Unsolicited Messages These messages are received whenever one of the following corresponding event occurs: sheet message on incoming calls, or updatestatus when a phone status changes.

sheet This message is received to display customer information if configured at the server side

```
{
    "timenow": 1361444639.61,
    "class": "sheet",
    "compressed": true,
    "serial": "xml",
    "payload": "AAADnnicndPBToNAEAbgV1n3XgFN1AP...................",
    "channel": "SIP/e6fhff-00000007"
}
```

How to decode payload :

```
>>> b64content = base64.b64decode(<payload content>)
>>> # 4 first cars are the encoded lenght of the xml string (in Big Endian format)
>>> xmllen = struck.unpack('>I',b64content[0:4])
>>> # the rest is a compressed xml string
>>> xmlcontent = zlib.decompress(toto[4:])
>>> print xmlcontent
```


```
<?xml version="1.0" encoding="utf-8"?>
   <profile>
      <user>
         <internal name="ipbxid"><![CDATA[xivo]]></internal>
         <internal name="where"><![CDATA[dial]]></internal>
         <internal name="channel"><![CDATA[SIP/barometrix_jyldev-00000009]]></internal>
         <internal name="focus"><![CDATA[no]]></internal>
         <internal name="zip"><![CDATA[1]]></internal>
         <sheet_qtui order="0010" name="qtui" type="None"><![CDATA[]]></sheet_qtui>
         <sheet_info order="0010" name="Nom" type="title"><![CDATA[0230210083]]></sheet_info>
         <sheet_info order="0030" name="Origine" type="text"><![CDATA[extern]]></sheet_info>
         <sheet_info order="0020" name="Num\xc3\xa9ro" type="text"><![CDATA[0230210083]]></sheet_
         <systray_info order="0010" name="Nom" type="title"><![CDATA[Maric\xc3\xa9 Sapr\xc3\xaftcl
         <systray_info order="0030" name="Origine" type="body"><![CDATA[extern]]></systray_info>
         <systray_info order="0020" name="Num\xc3\xa9ro" type="body"><![CDATA[0230210083]]></syst
      </user>
   </profile>
```


The xml file content is defined by the following xsd file: `xivo-javactilib/src/main/xsd/sheet.xsd` (online version)

phone status update Received when a phone status change

- class : getlist
- function : updatestatus
- listname : phones


```
{
   "class": "getlist",
   "function": "updatestatus",
   "listname": "phones",
   "tipbxid": "xivo",
   "timenow": 1361447017.29,
   .........
}
```


tid is the the object identification

Example of phone messages received when a phone is ringing :


```
{ ... "status": {"channels": ["SIP/x2gjtw-0000000b"]}, "tid": "3",}
{.... "status": {"channels": ["SIP/x2gjtw-0000000b"], "queues": [], "hintstatus": "0", "groups":
{.... "status": {"hintstatus": "8"}, "tid": "3"}
```


channel status update

- class : getlist
- function : updatestatus
- listname : channels
- status
  - direction : (in,out ...)

- state : (Down, Ring, Unknown ...)
- commstatus : (ready, calling, ringing ...)

```
{
   "class": "getlist",
   "function": "updatestatus",
   "listname": "channels",
   "tipbxid": "xivo",
   "timenow": 1361447017.29,
   .........
}
```

Example of phone messages received when a phone is ringing :

```
{"status": {"timestamp": 1361447017.22, "holded": false, "commstatus": "ready", "parked": false, 
{"status": {"timestamp": 1361447017.29, "holded": false, "commstatus": "ready", "parked": false, 
{"status": {"talkingto_kind": "channel", "direction": "out", "timestamp": 1361447017.29, "holded"
{"status": {"direction": "in", "timestamp": 1361447017.29, "holded": false, "talkingto_id": "SIP/
```

Configuration Messages The following messages are used to retrieve XiVO configuration.

Common fields

- class : getlist
- function : listid
- commandid
- tipbxid
- listname : Name of the list to be retreived : users, phones, agents, queues, voicemails, queuemembers

```
{
   "class": "getlist",
   "commandid": 489035169,
   "function": "listid",
   "tipbxid": "xivo",
   "listname": "........."
}
```

users Return a list of configured user id's

Client -> Server

```
{"class": "getlist", "commandid": 489035169, "function": "listid", "listname": "users", "tipbxid"
```

Server -> Client

```
{
   "class": "getlist",
   "function": "listid", "listname": "users",
   "list": ["11", "12", "14", "17", "1", "3", "2", "4", "9"],
   "tipbxid": "xivo","timenow": 1362735061.17
   }
```

user Return a user configuration

- tid is the userid returned by users message

Client -> Server

```
{
  "class": "getlist",
  "function": "updateconfig",
  "listname": "users",
  "tid": "17",
  "tpbxid": "xivo",  "commandid": 5}
```

```
Server -> Client
```

```
{
   "class": "getlist",
   "function": "updateconfig",
   "listname": "users",
   "tid": "17",
   "tipbxid": "xivo",
   "timenow": 1362741166.4,
   "config": {
         "enablednd": 0, "destrna": "", "enablerna": 0,  "enableunc": 0, "destunc": "", "destbusy
         "firstname": "Alice",  "lastname": "Bouzat", "fullname": "Alice Bouzat",
         "voicemailid": null, "incallfilter": 0,  "enablevoicemail": 0,  "profileclient": null, 
    }
```

phones `Client -> Server`

```
{"class": "getlist", "commandid": 495252308, "function": "listid", "listname": "phones", "tipbxid
```

```
Server > Client
```

```
{"class": "getlist", "function": "listid", "list": ["1", "3", "2", "5", "14", "7", "6", "9", "8"],
   "listname": "phones", "timenow": 1364994093.38, "tipbxid": "xivo"}
```

Individual phone configuration request:

```
{"class": "getlist", "commandid": 704096693, "function": "updateconfig", "listname": "phones", "t
```

```
Server > Client
```

```
{"class": "getlist",
   "config": {"allowtransfer": null, "context": "default", "identity": "SIP/ihvbur", "iduserfeatu
               "initialized": null, "number": "1000", "protocol": "sip"},
   "function": "updateconfig", "listname": "phones", "tid": "3", "timenow": 1364994093.43, "tipbx
```

agents `Client -> Server`

```
{"class": "getlist", "commandid": 1431355191, "function": "listid", "listname": "agents", "tipbxi
```

queues `Client -> Server`

```
{"class": "getlist", "commandid": 719950939, "function": "listid", "listname": "queues", "tipbxid
```

```
Server -> Client
```

```
{"function": "listid", "listname": "queues", "tipbxid": "xivo",
      "list": ["1", "10", "3", "2", "5", "4", "7", "6", "9", "8"], "timenow": 1382704649.64, "cla
```

queue tid is the id returned in the list field of the getlist response message

```
Client -> Server
```

```
{"commandid":7,"class":"getlist","tid":"3","tipbxid":"xivo","function":"updateconfig","listname":
```

Server -> Client

```
{
 "function": "updateconfig", "listname": "queues", "tipbxid": "xivo", "timenow": 1382704649.69, "t
   "config":
      {"displayname": "red", "name": "red", "context": "default", "number": "3002"},
 "class": "getlist"}
```

voicemails Client -> Server

```
{"class": "getlist", "commandid": 1034160761, "function": "listid", "listname": "voicemails", "ti
```

queuemembers Client -> Server

```
{"class": "getlist", "commandid": 964899043, "function": "listid", "listname": "queuemembers", "t
```

Server -> Client

```
{"function": "listid", "listname": "queuemembers", "tipbxid": "xivo",
   "list": ["Agent/2501,blue", "Agent/2500,yellow", "Agent/2002,yellow", "Agent/2003,__switchboar
            "Agent/2003,blue", "Agent/108,blue", "Agent/2002,blue"],
   "timenow": 1382717016.23,
   "class": "getlist"}
```

Status messages These messages can also be received without any request as unsolicited messages.

User status User status is to manage user presence

- Request user status update

Client -> Server

```
{"class": "getlist", "commandid": 107712156,
   "function": "updatestatus",
   "listname": "users",
   "tid": "14", "tipbxid": "xivo"}
```

Server > Client

```
{"class": "getlist",
   "function": "updatestatus",
   "listname": "users",
   "status": {"availstate": "outtolunch", "connection": "yes"},
         "tid": "1", "timenow": 1364994093.48, "tipbxid": "xivo"}
```

- Change User status

Client -> Server

```
{"availstate": "away",
    "class": "availstate",
    "commandid": 1946092392,
    "ipbxid": "xivo",
        "userid": "1"}
```

Server > Client

```
{"class": "getlist",
    "function": "updatestatus",
    "listname": "users",
    "status": {"availstate": "away"},
    "tid": "1", "timenow": 1370523352.6, "tipbxid": "xivo"}
```

Phone status

- tid is the line id, found in linelist from message user

`Client -> Server`

```
{"class": "getlist", "commandid": 107712156,
   "function": "updatestatus",
   "listname": "phones", "tid": "8", "tipbxid": "xivo"}
```

`Server > Client`

```
{"class": "getlist", "function": "updatestatus", "listname": "phones",
   "status": {"channels": [], "groups": [], "hintstatus": "0", "queues": []},
   "tid": "1", "timenow": 1364994093.48, "tipbxid": "xivo"}
```

Queue status

`Client -> Server`

```
{"commandid":17,"class":"getlist","tid":"8","tipbxid":"xivo","function":"updatestatus","listname"
```

`Server > Client`

```
{"function": "updatestatus", "listname": "queues", "tipbxid": "xivo", "timenow": 1382710430.54,
   "status": {"agentmembers": ["1","5"], "phonemembers": ["8"]},
   "tid": "8", "class": "getlist"}
```

Agent status

- tid is the agent id.

`Client -> Server`

```
{"class": "getlist",
 "commandid": <random_integer>,
 "function": "updatestatus",
 "listname": "agents",
 "tid": "635",
 "tipbxid": "xivo"}
```

`Server > Client`

```
{"class": "getlist",
 "listname": "agents",
 "function": "updatestatus",
 "tipbxid": "xivo",
 "tid": 635,
 "status": {
     "availability": "logged_out",
      "availability_since": 1370868774.74,
      "channel": null,
      "groups": [],
      "on_call_acd": false,
      "on_call_nonacd": false,
      "on_wrapup": false,
      "phonenumber": null,
      "queues": [
```

```
        "113"
    ]
}}
```

- availability can take the values:
  - logged_out
  - available
  - unavailable
  - on_call_nonacd_incoming_internal
  - on_call_nonacd_incoming_external
  - on_call_nonacd_outgoing_internal
  - on_call_nonacd_outgoing_external
- availability_since is the timestamp of the last availability change
- queues is the list of queue ids from which the agent receives calls

Agent messages

login `Client -> Server`

```
{"agentphonenumber": "1000", "class": "ipbxcommand", "command": "agentlogin", "commandid": 733366
```

agentphonenumber is the physical phone set where the agent is going to log on.

`Server > Client`

- Login successfull :

```
{"function": "updateconfig", "listname": "queuemembers", "tipbxid": "xivo",
   "timenow": 1362664323.94, "tid": "Agent/2002,blue",
   "config": {"paused": "0", "penalty": "0", "membership": "static", "status": "1", "lastcall": "
               "interface": "Agent/2002", "queue_name": "blue", "callstaken": "0"},
 "class": "getlist"
   }
```

```
{"function": "updatestatus", "listname": "agents", "tipbxid": "xivo",
   "timenow": 1362664323.94,
   "status": {"availability_since": 1362664323.94,
               "queues": [], "phonenumber": "1001", "on_call": false, "groups": [],
               "availability": "available", "channel": null},
   "tid": 7, "class": "getlist"
      }
```

- The phone number is already used by an other agent :

```
{"class": "ipbxcommand", "error_string": "agent_login_exten_in_use", "timenow": 1362664158.14}
```

Logout `Client -> Server`

```
{"class": "ipbxcommand", "command": "agentlogout", "commandid": 552759274}
```

Pause On all queues

`Client -> Server`

```
{"class": "ipbxcommand", "command": "queuepause", "commandid": 859140432, "member": "agent:xivo/1
```

Un pause On all queues

Client -> Server

```
{"class": "ipbxcommand", "command": "queueunpause", "commandid": 822604987, "member": "agent:xivo
```

Add an agent in a queue Client -> Server

```
{"class": "ipbxcommand", "command": "queueadd", "commandid": 542766213, "member": "agent:xivo/3",
```

Remove an agent from a queue Client -> Server

```
{"class": "ipbxcommand", "command": "queueremove", "commandid": 742480296, "member": "agent:xivo/
```

Listen to an agent Client -> Server

```
{"class": "ipbxcommand", "command": "listen", "commandid": 1423579492, "destination": "xivo/1", "
```

Service Messages

- class : featuresput

Call Filtering

- function : incallfilter
- value : true, false activate deactivate filtering

Client -> Server

```
{"class": "featuresput", "commandid": 1326845972, "function": "incallfilter", "value": true}
```

Server > Client

```
{
   "class": "getlist",
   "config": {"incallfilter": true},
   "function": "updateconfig",
   "listname": "users",
   "tid": "2",
   "timenow": 1361456398.52, "tipbxid": "xivo"   }
```

DND

- function : enablednd
- value : true, false activate deactivate DND

Client -> Server

```
{"class": "featuresput", "commandid": 1088978942, "function": "enablednd", "value": true}
```

Server > Client

```
{
   "class": "getlist",
   "config": {"enablednd": true},
   "function": "updateconfig",
   "listname": "users",
   "tid": "2",
   "timenow": 1361456614.55, "tipbxid": "xivo"}
```

Recording

- function : enablerecording
- value : true, false

Activate / deactivate recording for a user, extension call recording has to be activated : *Services->IPBX->IPBX services->Extension*

```
Client -> Server
```

```
{"class": "featuresput", "commandid": 1088978942, "function": "enablerecording", "value": true, "
```

```
Server > Client
```

```
{
   "class": "getlist",
   "config": {"enablerecording": true},
   "function": "updateconfig",
   "listname": "users",
   "tid": "7",
   "timenow": 1361456614.55, "tipbxid": "xivo"}
```

Unconditional Forward Forward the call at any time, call does not reach the user

- function : fwd

```
Client -> Server
```

```
{
   "class": "featuresput", "commandid": 2082138822, "function": "fwd",
   "value": {"destunc": "1002", "enableunc": true}
}
```

```
Server > Client
```

```
{
   "class": "getlist",
   "config": {"destunc": "1002", "enableunc": true},
   "function": "updateconfig",
   "listname": "users",
   "tid": "2",
   "timenow": 1361456777.98, "tipbxid": "xivo"}
```

Forward On No Answer Forward the call to another destination if the user does not answer

- function : fwd

```
Client -> Server
```

```
{
   "class": "featuresput", "commandid": 1705419982, "function": "fwd",
   "value": {"destrna": "1003", "enablerna": true}
   }
```

```
Server > Client
```

```
{
   "class": "getlist",
   "config": {"destrna": "1003", "enablerna": true},
   "function": "updateconfig",
   "listname": "users",
   "tid": "2",
   "timenow": 1361456966.89, "tipbxid": "xivo" }
```

Forward On Busy Forward the call to another destination when the user is busy

- function : fwd

```
Client -> Server
```

```
{
   "class": "featuresput", "commandid": 568274890, "function": "fwd",
   "value": {"destbusy": "1009", "enablebusy": true}
   }
```

```
Server > Client
```

```
{
   "class": "getlist",
   "config": {"destbusy": "1009", "enablebusy": true},
   "function": "updateconfig",
   "listname": "users",
   "tid": "2",
   "timenow": 1361457163.77, "tipbxid": "xivo"
   }
```

IPBX Commands

dial

- destination can be any number
- destination can be a pseudo URL of the form "type:ibpx/id"

```
Client -> Server
```

```
{
   "class": "ipbxcommand",
   "command": "dial",
   "commandid": <commandid>,
   "destination": "exten:xivo/<extension>"
}
```

For example :

```
{
    "class": "ipbxcommand",
    "command": "dial",
    "commandid": 1683305913,
    "destination": "exten:xivo/1202"
}
```

The server will answer with either an error or a success:

```
{
    "class": "ipbxcommand",
    "error_string": "unreachable_extension:1202",
```

```
}

{
    "class": "dial_success",
    "exten": "1202"
}
```

originate Same message than the dial message with a source fied. The source field is `user:xivo/<userid`, userid is replaced by a user identifer returned by the message getting users list

Example:

```
{
    "class": "ipbxcommand",
    "command": "originate",
    "commandid": 1683305913,
    "source":"user:xivo/34",
    "destination": "exten:xivo/1202"
}
```

record `Client -> Server`

- subcommand : `start` or `stop`

```
{
          "class": "ipbxcommand",
          "command": "record",
          "subcommand": "start",
          "channel": "SIP/x2gjtw-0000000d",
          "commandid": 1423579492
}
```

``` 
``Server > Client``
```

- response : `ok` request was correctly processed, `ko` unable to process the request

```
{"command": "record", "replyid": 1423579492, "class": "ipbxcommand", "ipbxreply": true, "timenow"
{"replyid": 1423579492, "command": "record", "class": "ipbxcommand", "timenow": 1361798879.13, "re
```

hangup `Client -> Server`

```
{
    "class": "ipbxcommand",
    "command": "hangup",
    "channelids": "chan:xivo/<channel_id>",
    "commandid": <command_id>
}
```

For example:

```
{
    "class": "ipbxcommand",
    "command": "hangup",
    "channelids": "chan:xivo/SIP/im2p7kzr-00000003",
    "commandid": 177773016
}
```

`Server -> Client`

```
{
    "class": "ipbxcommand",
    "command": "hangup",
```

```
    "ipbxreply": 1,
    "replyid": 177773016,
    "timenow": 1395756534.64
}
```

Statistics

subscribetoqueuesstats This message can be sent from the client to enable statitics update on queues

`Client -> Server`

```
{"commandid":36,"class":"subscribetoqueuesstats"}
```

``` ``Server > Client`` ```

getqueuesstats When statistic update is enable by sending message subscribetoqueuesstats.

The first element of the message is the queue id

```
{"stats": {"10": {"Xivo-LoggedAgents": 0}},
   "class": "getqueuesstats", "timenow": 1384509582.88}
{"stats": {"1": {"Xivo-WaitingCalls": 0}},
   "class": "getqueuesstats", "timenow": 1384509582.89}
{"stats": {"1": {"Xivo-TalkingAgents": "0", "Xivo-AvailableAgents": "1", "Xivo-EWT": "6"}},
   "class": "getqueuesstats", "timenow": 1384512350.25}
```

Switchboard

answer This allows the switchboard operator to answer an incoming call or unhold a call on-hold.

```
{"class": "answer", "uniqueid": "12345667.89"}
```

REGCOMMANDS

call_form_result This message is received when a *call form* is submitted from a client to the XiVO.

`Client -> Server`

```
{
    "class": "call_form_result",
    "commandid": <commandid>,
    "infos": {"buttonname": "saveandclose",
              "variables": {"XIVOFORM_varname1": "value1",
                            "XIVOFORM_varname2": "value2"}}
}
```

history

- **mode**
    - 0 : sent calls
    - 1 : received calls
    - 2 : missed calls
- size : Size of the list to be sent by the server
```

```
Client -> Server
```

```
{
   "mode": "0",
   "size": "8",
   "class": "history",
   "xuserid": "<xivoid>/<userfeaturesid>",
   "commandid": <commandid>
}
```

```
Server > Client
```

Send back a table of calls :

- duration in seconds

```
{
   "class": "history",
   "history": [
      {"calldate": "2013-03-29T08:44:35.273998", "duration": 0.148765, "fullname": "*844201"},
      {"calldate": "2013-03-28T16:56:48.071213", "duration": 58.134744, "fullname": "41400"}
   ],
   "mode": 0, "replyid": 529422441, "timenow": 1364571477.33
}
```

chitchat

```
{
   "class": "chitchat",
   "text": "message envoye",
   "to": "<xivoid>/<userfeaturesid>",
   "commandid": <commandid>
}
```

featuresget

featuresput

directory Request directory information, names matching pattern ignore case.

```
Client -> Server
```

```
{
   "class": "directory",
   "commandid": 1079140548,
   "pattern": "pau"
}
```

```
Server > Client
```

```
{
   "class": "directory",
   "headers": ["Nom", "Num\u00e9ro", "Mobile", "Autre num\u00e9ro", "E-mail", "Fonction", "Site",
   "replyid": 1079140548,
   "resultlist": ["Claire Mapaurtal;;+33644558899;31256;cmapaurtal@societe.com;;;",
                  "Paul Salvadier;+33445236988;+33678521430;31406;psalvadier@societe.com;;;"],
   "status": "ok",
   "timenow": 1378798928.26
}
```

parking

logfromclient

keepalive

availstate

filetransfer

faxsend

getipbxlist

```
{
    "class": "getipbxlist",
    "commandid": <commandid>
}
```

ipbxcommand

```
{
   "class": "ipbxcommand",
   "command": "originate",
   "commandid": <commandid>,
   "destination": "user:special:myvoicemail",
   "source": "user:special:me"
}
```

CTI server implementation

In the git repository `git://github.com/xivo-pbx/xivo-ctid.git`, under *xivo_ctid/*

- *cti_config* handles the configuration coming from the WEBI
- *interfaces/interface_ami*, together with *asterisk_ami_definitions*, *amiinterpret* and *xivo_ami* handle the AMI connections (asterisk)
- *interfaces/interface_fagi* handles the FAGI connections (still asterisk)
- *interfaces/interface_info* handles the CLI-like connections
- *interfaces/interface_webi* handles the requests and signals coming from the WEBI
- *interfaces/interface_cti* handles the clients' connections, with the help of *client_connection*, and it often involves *cti_command* too
- *interfaces/interface_rcti* handles the connections from the CTI server to other ones in the multi-xivo framework
- *innerdata* is meant to be the place where all statuses are computed and stored

The main loop uses *select()* syscall to dispatch the tasks according to miscellaneous incoming requests.

Requirements for *innerdata*:

- the properties fetched from the WEBI configuration shall be stored in the relevant *xod_config* structure
- the properties fetched from elsewhere shall be stored in the relevant *xod_status* structure
- at least two kinds of objects are not "predefined" (as are the phones or the queues, for instance)
  - the channels (in the asterisk SIP/345-0x12345678 meaning)
  - the group and queue members shall be handled in a special way each
  - most statuses of the calls should be set inside the channel structure

The purpose of the 'relations' field, in the various structures is to keep track of relations and cross-relations between different objects (a phone logged in as an agent, itself in a queue, itself called by some channels belonging to phones ...).

CTI server Message flow

Messages sent from the CTI clients to the server are received by the CTIServer class. The CTIServer then calls `interface_cti.CTI` class `manage_connection` method. The `interface_cti` uses his `_cti_command_handler` member to parse and run the command. The `CTICommandHandler` get a list of classes that handle this message from the `CTICommandFactory`. Then the the `interface_cti.CTI` calls `run_commands` on the handler, which returns a list of all commands replies.

To implement a new message in the protocol you have to create a new class that inherits the `CTICommand` class. Your new class should have a static member caller `required_fields` which is a list of required fields for this class. Your class should also have a `conditions` static member which is a list of tupples of conditions to detect that an incoming message matches this class. The `__init__` of your class is responsible for the initialization of it's fields and should call `super(<ClassName>, self).__init__(msg)`. Your class should register itself to the `CTICommandFactory`.

```
from xivo_cti.cti.cti_command import CTICommand
from xivo_cti.cti.cti_command_factory import CTICommandFactory

class InviteConfroom(CTICommand):
    required_fields = ['class', 'invitee']
    conditions = [('class', 'invite_confroom')]
    def __init__(self):
        super(InviteConfroom, self).__init__(msg)
        self._invitee = msg['invitee']

CTICommandFactory.register_class(InviteConfroom)
```

Each CTI commands has a callback list that you can register to from anywhere. Each callback function will be called when this message is received with the command as parameter.

Refer to `MeetmeList.__init__` for a callback registration example and to `MeetmeList.invite` for the implementation of a callback.

```
from xivo_cti.cti.commands.invite_confroom import InviteConfroom

class MySuperClass(object):
    def __init__(self):
        InviteConfroom.register_callback(self.invite_confroom_handler)

    def invite_confroom_handler(self, invite_confroom_command):
        # Do your stuff here.
        if ok:
            return invite_confroom_command.get_message('Everything is fine')
        else:
            return invite_confroom_command.get_warning('I don't know you, go away', True)
```

Note: The client's connection is injected in the command instance before calling callbacks functions. The client's connection is an `interface_cti.CTI` instance.

ppcticonf

ppcticonf is a small utility used to pretty print the CTI server configuration.

The utility is installed by default with XiVO. It comes with the xivo-utils package.

In fact, *ppcticonf* can be used to pretty print any URL or file that contains a JSON document.

How-to

The simplest invocation is:

```
$ ppcticonf
{
    "agentstatus": [],
    "bench": 0.077938079833984,
    "certfile": "/var/lib/xivo/certificates/test2.crt",
    "channelstatus": [],
    "contexts": {
        "*": {
            "didextens": {
                "*": [
                    "xivodir"
                ]
            }
        },
        "default": {
            "directories": [
                "xivodir",
                "internal"
            ],
            "display": "Display"
        }
   },
 ......
}
```

You can also pass a URL as an argument:

```
$ ppcticonf http://127.0.0.1/service/ipbx/json.php/private/pbx_settings/users
[
    {
        "agentid": null,
        "bsfilter": "no",
        "callerid": "\"User 2\"",
        "callrecord": false,
        "commented": false,
        "description": "",
        "destbusy": "",
        "destrna": "",
        "destunc": "",
        "enableautomon": false,
        "enablebusy": false,
        "enableclient": true,
        "enablednd": false,
        "enablehint": true,
        "enablerna": false,
        "enableunc": false,
        "enablevoicemail": false,
        "enablexfer": false,
        "entityid": 1,
        "firstname": "User",
        "fullname": "User 2",
        "id": 2,
        "identity": "User 2",
        "incallfilter": false,
 .....
]
```

Alternatively, you can pass a file path as an argument:

```
$ curl -s https://localhost/service/ipbx/json.php/private/ctiserver/configuration > cticonf
$ ppcticonf cticonf
{
    "agentstatus": [],
```

```
    "bench": 0.078904151916504003,
    "certfile": "/var/lib/xivo/certificates/test2.crt",
    "channelstatus": [],
    "contexts": {
        "*": {
            "didextens": {
                "*": [
                    "xivodir"
                ]
            }
        },
        "default": {
            "directories": [
                "xivodir",
                "internal"
            ],
            "display": "Display"
      }
.....
}
```

1.12.9 Database

Adding a Migration Script

Strating with XiVO 14.08, the database migration is handled by alembic.

The XiVO migration scripts can be found in the xivo-manage-db repository.

On a XiVO, they are located in the `/usr/share/xivo-manage-db` directory.

To add a new migration script from your developer machine, go into the root directory of the xivo-manage-db repository. There should be an `alembic.ini` file in this directory. You can then use the following command to create a new migration script:

```
alembic revision -m "<description>"
```

This will create a file in the `alembic/versions` directory, which you'll have to edit.

When the migration scripts are executed, they use a connection to the database with the role/user `asterisk`. This means that new objects that are created in the migration scripts will be owned by the `asterisk` role and it is thus not necessary (nor recommended) to explicitly grant access to objects to the asterisk role (i.e. no "GRANT ALL" command after a "CREATE TABLE" command).

1.12.10 Diagrams

Agent states

Graphs representing states and transitions between agent states. Used in Agent status dashboard and agent list.

`Download (DIA)`

1.12.11 Provisioning

This section describes the informations and tools for xivo-provd.

Managing DHCP server configuration

This page considers the configuration files of the DHCP server in `/etc/dhcp/dhcpd_update/`.

Who modifies the files

The files are updated with the command `dhcpd-update`, which is also run when updating the provisioning plugins. This commands fetches configurations files from the `provd.xivo.fr` server.

How to update the source files

Ensure your modifications are working

- On a XiVO, edit manually the file `/etc/dhcp/dhcpd_update/*.conf`
- `service isc-dhcp-server restart`
- If errors are shown in `/var/log/daemon.log`, check your modifications

Edit the files

- Edit the files in the Git repo `xivo-provd-plugins`, directory `dhcp/`
- Push your modifications
- Go in `dhcp/`
- Run `make upload` to push your modifications to `provd.xivo.fr`. There is no `testing` version of these files. Once the files are uploaded, they are available for all XiVO installations.

Managing Plugins

Git Repository

Most plugin-related files are available in the xivo-provd-plugins repository. Following examples are relative to the repository directory tree. Any modifications should be preceeded by a *git pull*.

Updating a Plugin

We will be using the *xivo-cisco-spa* plugins family as an example on this page

There is one directory per family. Here is the directory structure for `xivo-cisco-spa`:

```
plugins/xivo-cisco-spa/
+-- model_name_xxx
+-- model_name_xxx
+-- common
+-- build.py
```

Every plugin has a folder called `common` which regoups common ressources for each model. Every model has its own folder with its version number.

After modifying a plugin, you must increment the version number. You can modifiy the file `plugin-info` to change the version number:

```
plugins/xivo-cisco-spa/
+-- model_name_xxx
    +-- plugin-info
```

Important: If ever you modify the folder `common`, you must increment the version number of all the models.

Use Case: Update Firmwares for a given plugin Let us suppose we want to update firmwares for xivo-snom from 8.7.3.25 to 8.7.3.25 5. Here are the steps to follow :

1. Copy folder plugins/xivo-snom/8.7.3.25 to plugins/xivo-snom/8.7.3.25.5
2. Update VERSION number in plugins/xivo-snom/8.7.3.25.5/entry.py
3. Update VERSION number in plugins/xivo-snom/8.7.3.25.5/plugin-info
4. Download new firmwares (.bin files from snom website)
5. Update VERSION number and URIs in plugins/xivo-snom/8.7.3.25.5/pkgs/pkgs.db (with uris of downloaded files from snom website)
6. Update sizes and sha1sums in plugins/xivo-snom/8.7.3.25.5/pkgs/pkgs.db (using helper script xivo-tools/dev-tools/check_fw)
7. Update plugins/xivo-snom/build.py (duplicate and update section 8.7.3.25 > 8.7.3.25.5)

Test your changes You have three different methods to test your changes on your development machine.

Always increase plugin version (easiest) If the production version is 0.4, change the plugin version to 0.4.01, make your changes and upload to testing (see below).

Next modification will change the plugin version to 0.4.02, etc. When you are finished making changes, change the version to 0.5 and upload one last time.

Edit directly on XiVO Edit the files in `/var/lib/xivo-provd/plugins`.

To apply your changes, go in `provd_pycli` and run:

```
plugins.reload('xivo-cisco-spa-7.5.4')
```

Disable plugin caching Edit `/etc/xivo/provd/provd.conf` and add the line:

```
cache_plugin: True
```

Empty `/var/cache/xivo-provd` and restart provd.

Make your changes in provd-plugins, update the plugin version to the new one and upload to testing (see below). Now, every time you uninstall/install the plugin, the new plugin will be fetched from testing, instead of being cached, even without changing the version.

Uploading to testing Before updating a plugin, it must be passed through the testing phase. Once it has been approved it can be uploaded to the production server

Important: Before uploading a plugin in the testing provd repository, make sure to git pull the xivo-provd-plugins git repository.

To upload the modified plugin in the testing repo on *provd.xivo.fr*, you can execute the following command:

```
$ make upload
```

Afterwards, in the web-interface, you must modify the URL in section *Configuration → Provisioning → General* to:

```
`http://provd.xivo.fr/plugins/1/testing/`
```

You can then update the list of plugins and check the version number for the plugin that you modified. Don't forget to install the plugin to test it.

Mass-install all firmwares related to a given plugin Using provd_pycli on a xivo server, one can mass-install firmwares. Following example installs all firmwares for xivo-snom 8.7.3.25.5 plugin (note the auto-completion):

```
provpy> plugins.installed().keys()
[u'xivo-snom-8.7.3.15',
 u'xivo-cisco-sccp-legacy',
 u'xivo-snom-8.4.35',
 u'xivo-snom-8.7.3.25',
 u'xivo-aastra-switchboard',
 u'xivo-aastra-3.2.2-SP3',
 u'xivo-aastra-3.2.2.1136',
 u'xivo-cisco-sccp-9.0.3',
 u'null',
 u'xivo-snom-8.7.3.25.5']
provpy> p = plugins['xivo-snom-8.7.3.25.5']
provpy> p.install_all()
```

Uploading to stable Once checked, you must synchronize the plugin from *testing* to *stable*. If applicable, you should also update the archive repo.

To download the stable and archive plugins:

```
$ make download-stable
$ make download-archive
```

Go to the *plugins/_build* directory and delete the plugins that are going to be updated. Note that if you are not updating a plugin but you are instead removing it "once and for all", you should instead move it to the archive directory:

```
$ rm -fi stable/xivo-cisco-spa*
```

Copy the files from the directory *testing* to *stable*:

```
$ cp testing/xivo-cisco-spa* stable
```

Go back to the *plugins* directory and upload the files to the stable and archive repo:

```
$ make upload-stable
$ make upload-archive
```

The file are now up to date and you can test by putting back the *stable* url in the web-interface's configuration:

`'http://provd.xivo.fr/plugins/1/stable/'`

Testing a new SIP phone

Let's suppose you have received a brand new SIP phone that is not supported by the provisioning system of XiVO. You would like to know if it's possible to add auto-provisioning support for it. That said, you have never tested the phone before.

This guide will help you get through the different steps that are needed to add auto-provisioning support for a phone to XiVO.

Prerequisites

Before continuing, you'll need the following:

- a private LAN where only your phones and your test machines are connected to it, i.e. a LAN that you fully control.

Configuring a test environment

Although it's possible to do all the testing directly on a XiVO, it's more comfortable and usually easier to do on a separate, dedicated machine.

That said, you'll still need a XiVO near, since we'll be doing the call testing part on it and not on a separate asterisk.

So, for the rest of this guide, we'll suppose you are doing your tests on a *Debian Wheezy* with the following configuration:

- Installed packages:

  ```
  isc-dhcp-server tftpd-hpa apache2
  ```

- Example content of the `/etc/dhcp/dhcpd.conf` file (restart `isc-dhcp-server` after modification):

  ```
  ddns-update-style none;

  default-lease-time 7200;
  max-lease-time 86400;

  log-facility local7;

  subnet 10.34.1.0 netmask 255.255.255.0 {
      authoritative;

      range 10.34.1.200 10.34.1.250;

      option subnet-mask 255.255.255.0;
      option broadcast-address 10.34.1.255;
      option routers 10.34.1.6;

      option ntp-servers 10.34.1.6;
      option domain-name "my-domain.example.org";
      option domain-name-servers 10.34.1.6;

      log(concat("[VCI: ", option vendor-class-identifier, "]"));
  }
  ```

- Example content of the `/etc/default/tftpd-hpa` file (restart `tftpd-hpa` after modifcation):

  ```
  TFTP_USERNAME="tftp"
  TFTP_DIRECTORY="/srv/tftp"
  TFTP_ADDRESS="0.0.0.0:69"
  TFTP_OPTIONS="--secure --verbose"
  ```

With this configuration, files served via TFTP will be in the `/srv/tftp` directory and those served via HTTP in the `/var/www` directory.

Testing

Adding auto-provisioning support for a phone is mostly a question of finding answers to the following questions.

1. *Is it worth the time adding auto-provisioning support for the phone ?*

   Indeed. Adding quality auto-provisioning support for a phone to XiVO requires a non negligible amount of work, if you don't meet any real problem and are comfortable with provisioning in XiVO. Not all phones are born equal. Some are cheap. Some are old and slow. Some are made to work on proprietary system and will only work in degraded mode on anything else.

   That said, if you are uncertain, testing will help you clarifying your idea.

2. *What is the vendor, model, MAC address and firmware version (if available) of your phone ?*

   Having the vendor and model name is essential when looking for documentation or other information. The MAC address will be needed later on for some tests, and it's always good to know the firmware version of the phone if you are trying to upgrade to a newer firmware version and you're having some troubles, and when reading the documentation.

3. *Is the official administrator guide/documentation available publicly on the vendor web site ? Is it available only after registering and login to the vendor web site ?*

   Having access to the administrator guide/documentation of the phone is also essential. Once you've found it, download it and keep the link to the URL. If you can't find it, it's probably not worth going further.

4. *Is the latest firmware of the phone available publicly on the vendor web site ? Is it available only after registering and login to the vendor web site ?*

   Good auto-provisioning support means you need to have an easy way to download the latest firmware of the phone. Ideally, this mean the firmware is downloadable from an URL, with no authentication whatsoever. In the worst case, you'll need to login on some web portal before being able to download the firmware, which will be cumbersome to automatize and probably fragile. If this is the case, it's probably not worth going further.

5. *Does the phone need other files, like language files ? If so, are these files available publicly on the vendor web site ? After registering ?*

   Although you might not be able to answer to this question yet because you might not know if the phone needs such files to be either in English or in French (the two officially supported language in XiVO), you'll need to have an easy access to these files if its the case.

6. *Does the phone supports auto-provisioning via DHCP + HTTP (or TFTP) ?*

   The provisioning system in XiVO is based on the popular method of using a DHCP server to tell the phone where to download its configuration files, and a HTTP (or TFTP) server to serve these configuration files. Some phones support other methods of provisioning (like TR-069), but that's of no use here. Also, if your phone is only configurable via its web interface, although it's technically possible to configure it automatically by navigating its web interface, it's an **extremely bad** idea since it's impossible to guarantee that you'll still be able to provision the phone on the next firmware release.

   If the phone supports both HTTP and TFTP, pick HTTP, it usually works better with the provisioning server of XiVO.

7. *What are the default usernames/passwords on the phone to access administrator menus (phone UI and web UI) ? How do you do a factory reset of the phone ?*

   Although this step is optional, it might be handy later to have these kind of information. Try to find them now, and note them somewhere.

8. *What are the DHCP options and their values to send to the phones to tell it where its configuration files are located ?*

   Once you know that the phone supports DHCP + HTTP provisioning, the next question is what do you need to put in the DHCP response to tell the phone where its configuration files are located. Unless the admin documentation of the phone is really poor, this should not be too hard to find.

   Once you have found this information, the easiest way to send it to the phone is to create a custom host declaration for the phone in the `/etc/dhcp/dhcpd.conf` file, like in this example:

   ```
   host my-phone {
      hardware ethernet 00:11:22:33:44:55;
      option tftp-server-name "http://169.254.0.1/foobar.cfg";
   }
   ```

9. *What are the configuration files the phone needs (filename and content) and what do we need to put in it for the phone to minimally be able to make and receive calls on XiVO ?*

Now that you are able to tell your phone where to look for its configuration files, you need to write these files with the right content in it. Again, at this step, you'll need to look through the documentation or examples to answer this question.

Note that you only want to have the most basic configuration here, i.e. only configure 1 line, with the right SIP registrar and proxy, and the associated username and password.

10. *Do basic telephony services, like transfer, works correctly when using the phone buttons ?*

    On most phones, it's possible to do transfer (both attended and direct), three-way conferences or put someone on hold directly from the phone. Do some tests to see if it works correctly.

    Also at this step, it's a good idea to check how the phone handle non-ascii characters, either in the caller ID or in its configuration files.

11. *Does other "standard" features work correctly on the phone ?*

    For quality auto-provisioning support, you must find how to configure and make the following features work:

    - NTP server
    - MWI
    - function keys (speed dial, BLF, directed pickup / call interception)
    - timezone and DST support
    - multi language
    - DTMF
    - hard keys, like the voicemail hard key on some phone
    - non-ASCII labels (line name, function key label)
    - non-ASCII caller ID
    - backup proxy/registrar
    - paging

Once you have answered all these questions, you'll have a good idea on how the phone works and how to configure it. Next step would be to start the development of a new provd plugin for your phone for a specific firmware version.

1.12.12 SCCP

Introduction

SCCP (or skinny) is a stimulus protocol used to fully interact with Cisco phones.

What is xivo-libsccp ? It's a SCCP channel driver written for Asterisk by Avencall based on the channel skinny.

Installation

The following packages are required to compile xivo-libsccp on a XiVO.

- build-essential
- asterisk-dev

```
apt-get update && apt-get install build-essential asterisk-dev

git clone https://github.com/xivo-pbx/xivo-libsccp.git
cd xivo-libsccp/xivo-libsccp/
make
make install
```

Configuration

See sccp.conf.sample for a configuration file example.

FAQ

```
Q. When is this *feature X* will be available?
A. The order in which we implement features is based on our client needs. Write
   us an email that clearly explain your setup and what you would like to do and we
   will see what we can do. We don't provide any timeline.

Q. I want to use the Page() application to call many phones at the same time.
A. Here a Page() example for a one way call (half-duplex):

exten => 1000,1,Verbose(2, Paging to external cisco phone)
same => n,Page(sccp/100/autoanswer&sccp/101/autoanswer,i,120 )

...for a two-way call (full-duplex):

exten => 1000,1,Verbose(2, Paging to external cisco phone)
same => n,Page(sccp/100/autoanswer&sccp/101/autoanswer,di,120 )
```

Network Configuration for 7920/7921

Here's how to to configure a hostapd based AP on a Debian host so that both a 7920 and 7921 Wi-Fi phone can connect to it.

The 7920 is older than the 7921 and is pretty limited in its Wi-Fi functionnality:

- 802.11b
- WPA (no WPA2)
- TKIP (no CCMP/AES)

Which means that the most secure WLAN you can set up if you want both phones to connect to it is not that secure.

1. Make sure you have a wireless NIC capable of master mode.
2. If needed, install the firmware-<vendor> package. For example, if you have a ralink card like I do:

   ```
   apt-get install firmware-ralink
   ```

3. Install the other dependencies:

   ```
   apt-get install wireless-tools hostapd bridge-utils
   ```

4. Create an hostapd configuration file in `/etc/hostapd/hostapd.sccp.conf` with content:

   ```
   ##### hostapd configuration file ##############################################
   # Empty lines and lines starting with # are ignored

   # AP netdevice name (without 'ap' postfix, i.e., wlan0 uses wlan0ap for
   # management frames); ath0 for madwifi
   interface=wlan0

   # In case of madwifi, atheros, and nl80211 driver interfaces, an additional
   # configuration parameter, bridge, may be used to notify hostapd if the
   # interface is included in a bridge. This parameter is not used with Host AP
   # driver. If the bridge parameter is not set, the drivers will automatically
   # figure out the bridge interface (assuming sysfs is enabled and mounted to
   # /sys) and this parameter may not be needed.
   ```

```
#
# For nl80211, this parameter can be used to request the AP interface to be
# added to the bridge automatically (brctl may refuse to do this before hostapd
# has been started to change the interface mode). If needed, the bridge
# interface is also created.
#bridge=br0

# Driver interface type (hostap/wired/madwifi/test/none/nl80211/bsd);
# default: hostap). nl80211 is used with all Linux mac80211 drivers.
# Use driver=none if building hostapd as a standalone RADIUS server that does
# not control any wireless/wired driver.
# driver=hostap

# hostapd event logger configuration
#
# Two output method: syslog and stdout (only usable if not forking to
# background).
#
# Module bitfield (ORed bitfield of modules that will be logged; -1 = all
# modules):
# bit 0 (1) = IEEE 802.11
# bit 1 (2) = IEEE 802.1X
# bit 2 (4) = RADIUS
# bit 3 (8) = WPA
# bit 4 (16) = driver interface
# bit 5 (32) = IAPP
# bit 6 (64) = MLME
#
# Levels (minimum value for logged events):
#  0 = verbose debugging
#  1 = debugging
#  2 = informational messages
#  3 = notification
#  4 = warning
#
logger_syslog=-1
logger_syslog_level=2
logger_stdout=-1
logger_stdout_level=2

# Dump file for state information (on SIGUSR1)
dump_file=/tmp/hostapd.dump

# Interface for separate control program. If this is specified, hostapd
# will create this directory and a UNIX domain socket for listening to requests
# from external programs (CLI/GUI, etc.) for status information and
# configuration. The socket file will be named based on the interface name, so
# multiple hostapd processes/interfaces can be run at the same time if more
# than one interface is used.
# /var/run/hostapd is the recommended directory for sockets and by default,
# hostapd_cli will use it when trying to connect with hostapd.
ctrl_interface=/var/run/hostapd

# Access control for the control interface can be configured by setting the
# directory to allow only members of a group to use sockets. This way, it is
# possible to run hostapd as root (since it needs to change network
# configuration and open raw sockets) and still allow GUI/CLI components to be
# run as non-root users. However, since the control interface can be used to
# change the network configuration, this access needs to be protected in many
# cases. By default, hostapd is configured to use gid 0 (root). If you
# want to allow non-root users to use the contron interface, add a new group
# and change this value to match with that group. Add users that should have
# control interface access to this group.
```

```
#
# This variable can be a group name or gid.
#ctrl_interface_group=wheel
ctrl_interface_group=0


##### IEEE 802.11 related configuration #######################################

# SSID to be used in IEEE 802.11 management frames
ssid=example-ssid

# Country code (ISO/IEC 3166-1). Used to set regulatory domain.
# Set as needed to indicate country in which device is operating.
# This can limit available channels and transmit power.
country_code=CA

# Enable IEEE 802.11d. This advertises the country_code and the set of allowed
# channels and transmit power levels based on the regulatory limits. The
# country_code setting must be configured with the correct country for
# IEEE 802.11d functions.
# (default: 0 = disabled)
#ieee80211d=1

# Operation mode (a = IEEE 802.11a, b = IEEE 802.11b, g = IEEE 802.11g,
# Default: IEEE 802.11b
# 7920 only supports b
hw_mode=b

# Channel number (IEEE 802.11)
# (default: 0, i.e., not set)
# Please note that some drivers do not use this value from hostapd and the
# channel will need to be configured separately with iwconfig.
channel=5

# Beacon interval in kus (1.024 ms) (default: 100; range 15..65535)
beacon_int=100

# DTIM (delivery traffic information message) period (range 1..255):
# number of beacons between DTIMs (1 = every beacon includes DTIM element)
# (default: 2)
dtim_period=2

# Maximum number of stations allowed in station table. New stations will be
# rejected after the station table is full. IEEE 802.11 has a limit of 2007
# different association IDs, so this number should not be larger than that.
# (default: 2007)
max_num_sta=255

# RTS/CTS threshold; 2347 = disabled (default); range 0..2347
# If this field is not included in hostapd.conf, hostapd will not control
# RTS threshold and 'iwconfig wlan# rts <val>' can be used to set it.
rts_threshold=2347

# Fragmentation threshold; 2346 = disabled (default); range 256..2346
# If this field is not included in hostapd.conf, hostapd will not control
# fragmentation threshold and 'iwconfig wlan# frag <val>' can be used to set
# it.
fragm_threshold=2346

# Rate configuration
# Default is to enable all rates supported by the hardware. This configuration
# item allows this list be filtered so that only the listed rates will be left
# in the list. If the list is empty, all rates are used. This list can have
```

```
# entries that are not in the list of rates the hardware supports (such entries
# are ignored). The entries in this list are in 100 kbps, i.e., 11 Mbps = 110.
# If this item is present, at least one rate have to be matching with the rates
# hardware supports.
# default: use the most common supported rate setting for the selected
# hw_mode (i.e., this line can be removed from configuration file in most
# cases)
#supported_rates=10 20 55 110 60 90 120 180 240 360 480 540

# Basic rate set configuration
# List of rates (in 100 kbps) that are included in the basic rate set.
# If this item is not included, usually reasonable default set is used.
#basic_rates=10 20
#basic_rates=10 20 55 110
#basic_rates=60 120 240

# Short Preamble
# This parameter can be used to enable optional use of short preamble for
# frames sent at 2 Mbps, 5.5 Mbps, and 11 Mbps to improve network performance.
# This applies only to IEEE 802.11b-compatible networks and this should only be
# enabled if the local hardware supports use of short preamble. If any of the
# associated STAs do not support short preamble, use of short preamble will be
# disabled (and enabled when such STAs disassociate) dynamically.
# 0 = do not allow use of short preamble (default)
# 1 = allow use of short preamble
#preamble=1

# Station MAC address -based authentication
# Please note that this kind of access control requires a driver that uses
# hostapd to take care of management frame processing and as such, this can be
# used with driver=hostap or driver=nl80211, but not with driver=madwifi.
# 0 = accept unless in deny list
# 1 = deny unless in accept list
# 2 = use external RADIUS server (accept/deny lists are searched first)
macaddr_acl=0

# Accept/deny lists are read from separate files (containing list of
# MAC addresses, one per line). Use absolute path name to make sure that the
# files can be read on SIGHUP configuration reloads.
#accept_mac_file=/etc/hostapd.accept
#deny_mac_file=/etc/hostapd.deny

# IEEE 802.11 specifies two authentication algorithms. hostapd can be
# configured to allow both of these or only one. Open system authentication
# should be used with IEEE 802.1X.
# Bit fields of allowed authentication algorithms:
# bit 0 = Open System Authentication
# bit 1 = Shared Key Authentication (requires WEP)
auth_algs=1

# Send empty SSID in beacons and ignore probe request frames that do not
# specify full SSID, i.e., require stations to know SSID.
# default: disabled (0)
# 1 = send empty (length=0) SSID in beacon and ignore probe request for
#     broadcast SSID
# 2 = clear SSID (ASCII 0), but keep the original length (this may be required
#     with some clients that do not support empty SSID) and ignore probe
#     requests for broadcast SSID
ignore_broadcast_ssid=0

# TX queue parameters (EDCF / bursting)
# tx_queue_<queue name>_<param>
# queues: data0, data1, data2, data3, after_beacon, beacon
```

```
#          (data0 is the highest priority queue)
# parameters:
#   aifs: AIFS (default 2)
#   cwmin: cwMin (1, 3, 7, 15, 31, 63, 127, 255, 511, 1023)
#   cwmax: cwMax (1, 3, 7, 15, 31, 63, 127, 255, 511, 1023); cwMax >= cwMin
#   burst: maximum length (in milliseconds with precision of up to 0.1 ms) for
#          bursting
#
# Default WMM parameters (IEEE 802.11 draft; 11-03-0504-03-000e):
# These parameters are used by the access point when transmitting frames
# to the clients.
#
# Low priority / AC_BK = background
#tx_queue_data3_aifs=7
#tx_queue_data3_cwmin=15
#tx_queue_data3_cwmax=1023
#tx_queue_data3_burst=0
# Note: for IEEE 802.11b mode: cWmin=31 cWmax=1023 burst=0
#
# Normal priority / AC_BE = best effort
#tx_queue_data2_aifs=3
#tx_queue_data2_cwmin=15
#tx_queue_data2_cwmax=63
#tx_queue_data2_burst=0
# Note: for IEEE 802.11b mode: cWmin=31 cWmax=127 burst=0
#
# High priority / AC_VI = video
#tx_queue_data1_aifs=1
#tx_queue_data1_cwmin=7
#tx_queue_data1_cwmax=15
#tx_queue_data1_burst=3.0
# Note: for IEEE 802.11b mode: cWmin=15 cWmax=31 burst=6.0
#
# Highest priority / AC_VO = voice
#tx_queue_data0_aifs=1
#tx_queue_data0_cwmin=3
#tx_queue_data0_cwmax=7
#tx_queue_data0_burst=1.5
# Note: for IEEE 802.11b mode: cWmin=7 cWmax=15 burst=3.3

# 802.1D Tag (= UP) to AC mappings
# WMM specifies following mapping of data frames to different ACs. This mapping
# can be configured using Linux QoS/tc and sch_pktpri.o module.
# 802.1D Tag      802.1D Designation      Access Category WMM Designation
# 1               BK                      AC_BK           Background
# 2               -                       AC_BK           Background
# 0               BE                      AC_BE           Best Effort
# 3               EE                      AC_BE           Best Effort
# 4               CL                      AC_VI           Video
# 5               VI                      AC_VI           Video
# 6               VO                      AC_VO           Voice
# 7               NC                      AC_VO           Voice
# Data frames with no priority information: AC_BE
# Management frames: AC_VO
# PS-Poll frames: AC_BE

# Default WMM parameters (IEEE 802.11 draft; 11-03-0504-03-000e):
# for 802.11a or 802.11g networks
# These parameters are sent to WMM clients when they associate.
# The parameters will be used by WMM clients for frames transmitted to the
# access point.
#
# note - txop_limit is in units of 32microseconds
```

```
# note - acm is admission control mandatory flag. 0 = admission control not
# required, 1 = mandatory
# note - here cwMin and cmMax are in exponent form. the actual cw value used
# will be (2^n)-1 where n is the value given here
#
wmm_enabled=1
#
# WMM-PS Unscheduled Automatic Power Save Delivery [U-APSD]
# Enable this flag if U-APSD supported outside hostapd (eg., Firmware/driver)
#uapsd_advertisement_enabled=1
#
# Low priority / AC_BK = background
wmm_ac_bk_cwmin=4
wmm_ac_bk_cwmax=10
wmm_ac_bk_aifs=7
wmm_ac_bk_txop_limit=0
wmm_ac_bk_acm=0
# Note: for IEEE 802.11b mode: cWmin=5 cWmax=10
#
# Normal priority / AC_BE = best effort
wmm_ac_be_aifs=3
wmm_ac_be_cwmin=4
wmm_ac_be_cwmax=10
wmm_ac_be_txop_limit=0
wmm_ac_be_acm=0
# Note: for IEEE 802.11b mode: cWmin=5 cWmax=7
#
# High priority / AC_VI = video
wmm_ac_vi_aifs=2
wmm_ac_vi_cwmin=3
wmm_ac_vi_cwmax=4
wmm_ac_vi_txop_limit=94
wmm_ac_vi_acm=0
# Note: for IEEE 802.11b mode: cWmin=4 cWmax=5 txop_limit=188
#
# Highest priority / AC_VO = voice
wmm_ac_vo_aifs=2
wmm_ac_vo_cwmin=2
wmm_ac_vo_cwmax=3
wmm_ac_vo_txop_limit=47
wmm_ac_vo_acm=0
# Note: for IEEE 802.11b mode: cWmin=3 cWmax=4 burst=102

# Static WEP key configuration
#
# The key number to use when transmitting.
# It must be between 0 and 3, and the corresponding key must be set.
# default: not set
#wep_default_key=0
# The WEP keys to use.
# A key may be a quoted string or unquoted hexadecimal digits.
# The key length should be 5, 13, or 16 characters, or 10, 26, or 32
# digits, depending on whether 40-bit (64-bit), 104-bit (128-bit), or
# 128-bit (152-bit) WEP is used.
# Only the default key must be supplied; the others are optional.
# default: not set
#wep_key0=123456789a
#wep_key1="vwxyz"
#wep_key2=0102030405060708090a0b0c0d
#wep_key3=".2.4.6.8.0.23"

# Station inactivity limit
#
```

```
# If a station does not send anything in ap_max_inactivity seconds, an
# empty data frame is sent to it in order to verify whether it is
# still in range. If this frame is not ACKed, the station will be
# disassociated and then deauthenticated. This feature is used to
# clear station table of old entries when the STAs move out of the
# range.
#
# The station can associate again with the AP if it is still in range;
# this inactivity poll is just used as a nicer way of verifying
# inactivity; i.e., client will not report broken connection because
# disassociation frame is not sent immediately without first polling
# the STA with a data frame.
# default: 300 (i.e., 5 minutes)
#ap_max_inactivity=300

# Disassociate stations based on excessive transmission failures or other
# indications of connection loss. This depends on the driver capabilities and
# may not be available with all drivers.
#disassoc_low_ack=1

# Maximum allowed Listen Interval (how many Beacon periods STAs are allowed to
# remain asleep). Default: 65535 (no limit apart from field size)
#max_listen_interval=100

# WDS (4-address frame) mode with per-station virtual interfaces
# (only supported with driver=nl80211)
# This mode allows associated stations to use 4-address frames to allow layer 2
# bridging to be used.
#wds_sta=1

# If bridge parameter is set, the WDS STA interface will be added to the same
# bridge by default. This can be overridden with the wds_bridge parameter to
# use a separate bridge.
#wds_bridge=wds-br0

# Client isolation can be used to prevent low-level bridging of frames between
# associated stations in the BSS. By default, this bridging is allowed.
#ap_isolate=1

##### IEEE 802.11n related configuration ######################################

# ieee80211n: Whether IEEE 802.11n (HT) is enabled
# 0 = disabled (default)
# 1 = enabled
# Note: You will also need to enable WMM for full HT functionality.
#ieee80211n=1

# ht_capab: HT capabilities (list of flags)
# LDPC coding capability: [LDPC] = supported
# Supported channel width set: [HT40-] = both 20 MHz and 40 MHz with secondary
# channel below the primary channel; [HT40+] = both 20 MHz and 40 MHz
# with secondary channel below the primary channel
# (20 MHz only if neither is set)
# Note: There are limits on which channels can be used with HT40- and
# HT40+. Following table shows the channels that may be available for
# HT40- and HT40+ use per IEEE 802.11n Annex J:
# freq          HT40-           HT40+
# 2.4 GHz       5-13            1-7 (1-9 in Europe/Japan)
# 5 GHz         40,48,56,64     36,44,52,60
# (depending on the location, not all of these channels may be available
# for use)
# Please note that 40 MHz channels may switch their primary and secondary
# channels if needed or creation of 40 MHz channel maybe rejected based
```

```
# on overlapping BSSes. These changes are done automatically when hostapd
# is setting up the 40 MHz channel.
# Spatial Multiplexing (SM) Power Save: [SMPS-STATIC] or [SMPS-DYNAMIC]
# (SMPS disabled if neither is set)
# HT-greenfield: [GF] (disabled if not set)
# Short GI for 20 MHz: [SHORT-GI-20] (disabled if not set)
# Short GI for 40 MHz: [SHORT-GI-40] (disabled if not set)
# Tx STBC: [TX-STBC] (disabled if not set)
# Rx STBC: [RX-STBC1] (one spatial stream), [RX-STBC12] (one or two spatial
# streams), or [RX-STBC123] (one, two, or three spatial streams); Rx STBC
# disabled if none of these set
# HT-delayed Block Ack: [DELAYED-BA] (disabled if not set)
# Maximum A-MSDU length: [MAX-AMSDU-7935] for 7935 octets (3839 octets if not
# set)
# DSSS/CCK Mode in 40 MHz: [DSSS_CCK-40] = allowed (not allowed if not set)
# PSMP support: [PSMP] (disabled if not set)
# L-SIG TXOP protection support: [LSIG-TXOP-PROT] (disabled if not set)
#ht_capab=[HT40-][SHORT-GI-20][SHORT-GI-40]

# Require stations to support HT PHY (reject association if they do not)
#require_ht=1

##### IEEE 802.1X-2004 related configuration ##################################

# Require IEEE 802.1X authorization
#ieee8021x=1

# IEEE 802.1X/EAPOL version
# hostapd is implemented based on IEEE Std 802.1X-2004 which defines EAPOL
# version 2. However, there are many client implementations that do not handle
# the new version number correctly (they seem to drop the frames completely).
# In order to make hostapd interoperate with these clients, the version number
# can be set to the older version (1) with this configuration value.
#eapol_version=2

# Optional displayable message sent with EAP Request-Identity. The first \0
# in this string will be converted to ASCII-0 (nul). This can be used to
# separate network info (comma separated list of attribute=value pairs); see,
# e.g., RFC 4284.
#eap_message=hello
#eap_message=hello\0networkid=netw,nasid=foo,portid=0,NAIRealms=example.com

# WEP rekeying (disabled if key lengths are not set or are set to 0)
# Key lengths for default/broadcast and individual/unicast keys:
# 5 = 40-bit WEP (also known as 64-bit WEP with 40 secret bits)
# 13 = 104-bit WEP (also known as 128-bit WEP with 104 secret bits)
#wep_key_len_broadcast=5
#wep_key_len_unicast=5
# Rekeying period in seconds. 0 = do not rekey (i.e., set keys only once)
#wep_rekey_period=300

# EAPOL-Key index workaround (set bit7) for WinXP Supplicant (needed only if
# only broadcast keys are used)
eapol_key_index_workaround=0

# EAP reauthentication period in seconds (default: 3600 seconds; 0 = disable
# reauthentication).
#eap_reauth_period=3600

# Use PAE group address (01:80:c2:00:00:03) instead of individual target
# address when sending EAPOL frames with driver=wired. This is the most common
# mechanism used in wired authentication, but it also requires that the port
# is only used by one station.
```

```
#use_pae_group_addr=1

##### Integrated EAP server ###################################################

# Optionally, hostapd can be configured to use an integrated EAP server
# to process EAP authentication locally without need for an external RADIUS
# server. This functionality can be used both as a local authentication server
# for IEEE 802.1X/EAPOL and as a RADIUS server for other devices.

# Use integrated EAP server instead of external RADIUS authentication
# server. This is also needed if hostapd is configured to act as a RADIUS
# authentication server.
eap_server=0

# Path for EAP server user database
#eap_user_file=/etc/hostapd.eap_user

# CA certificate (PEM or DER file) for EAP-TLS/PEAP/TTLS
#ca_cert=/etc/hostapd.ca.pem

# Server certificate (PEM or DER file) for EAP-TLS/PEAP/TTLS
#server_cert=/etc/hostapd.server.pem

# Private key matching with the server certificate for EAP-TLS/PEAP/TTLS
# This may point to the same file as server_cert if both certificate and key
# are included in a single file. PKCS#12 (PFX) file (.p12/.pfx) can also be
# used by commenting out server_cert and specifying the PFX file as the
# private_key.
#private_key=/etc/hostapd.server.prv

# Passphrase for private key
#private_key_passwd=secret passphrase

# Enable CRL verification.
# Note: hostapd does not yet support CRL downloading based on CDP. Thus, a
# valid CRL signed by the CA is required to be included in the ca_cert file.
# This can be done by using PEM format for CA certificate and CRL and
# concatenating these into one file. Whenever CRL changes, hostapd needs to be
# restarted to take the new CRL into use.
# 0 = do not verify CRLs (default)
# 1 = check the CRL of the user certificate
# 2 = check all CRLs in the certificate path
#check_crl=1

# dh_file: File path to DH/DSA parameters file (in PEM format)
# This is an optional configuration file for setting parameters for an
# ephemeral DH key exchange. In most cases, the default RSA authentication does
# not use this configuration. However, it is possible setup RSA to use
# ephemeral DH key exchange. In addition, ciphers with DSA keys always use
# ephemeral DH keys. This can be used to achieve forward secrecy. If the file
# is in DSA parameters format, it will be automatically converted into DH
# params. This parameter is required if anonymous EAP-FAST is used.
# You can generate DH parameters file with OpenSSL, e.g.,
# "openssl dhparam -out /etc/hostapd.dh.pem 1024"
#dh_file=/etc/hostapd.dh.pem

# Fragment size for EAP methods
#fragment_size=1400

# Configuration data for EAP-SIM database/authentication gateway interface.
# This is a text string in implementation specific format. The example
# implementation in eap_sim_db.c uses this as the UNIX domain socket name for
# the HLR/AuC gateway (e.g., hlr_auc_gw). In this case, the path uses "unix:"
```

```
# prefix.
#eap_sim_db=unix:/tmp/hlr_auc_gw.sock

# Encryption key for EAP-FAST PAC-Opaque values. This key must be a secret,
# random value. It is configured as a 16-octet value in hex format. It can be
# generated, e.g., with the following command:
# od -tx1 -v -N16 /dev/random | colrm 1 8 | tr -d ' '
#pac_opaque_encr_key=000102030405060708090a0b0c0d0e0f

# EAP-FAST authority identity (A-ID)
# A-ID indicates the identity of the authority that issues PACs. The A-ID
# should be unique across all issuing servers. In theory, this is a variable
# length field, but due to some existing implementations requiring A-ID to be
# 16 octets in length, it is strongly recommended to use that length for the
# field to provid interoperability with deployed peer implementations. This
# field is configured in hex format.
#eap_fast_a_id=101112131415161718191a1b1c1d1e1f

# EAP-FAST authority identifier information (A-ID-Info)
# This is a user-friendly name for the A-ID. For example, the enterprise name
# and server name in a human-readable format. This field is encoded as UTF-8.
#eap_fast_a_id_info=test server

# Enable/disable different EAP-FAST provisioning modes:
#0 = provisioning disabled
#1 = only anonymous provisioning allowed
#2 = only authenticated provisioning allowed
#3 = both provisioning modes allowed (default)
#eap_fast_prov=3

# EAP-FAST PAC-Key lifetime in seconds (hard limit)
#pac_key_lifetime=604800

# EAP-FAST PAC-Key refresh time in seconds (soft limit on remaining hard
# limit). The server will generate a new PAC-Key when this number of seconds
# (or fewer) of the lifetime remains.
#pac_key_refresh_time=86400

# EAP-SIM and EAP-AKA protected success/failure indication using AT_RESULT_IND
# (default: 0 = disabled).
#eap_sim_aka_result_ind=1

# Trusted Network Connect (TNC)
# If enabled, TNC validation will be required before the peer is allowed to
# connect. Note: This is only used with EAP-TTLS and EAP-FAST. If any other
# EAP method is enabled, the peer will be allowed to connect without TNC.
#tnc=1


##### IEEE 802.11f - Inter-Access Point Protocol (IAPP) #######################

# Interface to be used for IAPP broadcast packets
#iapp_interface=eth0


##### RADIUS client configuration #############################################
# for IEEE 802.1X with external Authentication Server, IEEE 802.11
# authentication with external ACL for MAC addresses, and accounting

# The own IP address of the access point (used as NAS-IP-Address)
own_ip_addr=127.0.0.1

# Optional NAS-Identifier string for RADIUS messages. When used, this should be
```

```
# a unique to the NAS within the scope of the RADIUS server. For example, a
# fully qualified domain name can be used here.
# When using IEEE 802.11r, nas_identifier must be set and must be between 1 and
# 48 octets long.
#nas_identifier=ap.example.com

# RADIUS authentication server
#auth_server_addr=127.0.0.1
#auth_server_port=1812
#auth_server_shared_secret=secret

# RADIUS accounting server
#acct_server_addr=127.0.0.1
#acct_server_port=1813
#acct_server_shared_secret=secret

# Secondary RADIUS servers; to be used if primary one does not reply to
# RADIUS packets. These are optional and there can be more than one secondary
# server listed.
#auth_server_addr=127.0.0.2
#auth_server_port=1812
#auth_server_shared_secret=secret2
#
#acct_server_addr=127.0.0.2
#acct_server_port=1813
#acct_server_shared_secret=secret2

# Retry interval for trying to return to the primary RADIUS server (in
# seconds). RADIUS client code will automatically try to use the next server
# when the current server is not replying to requests. If this interval is set,
# primary server will be retried after configured amount of time even if the
# currently used secondary server is still working.
#radius_retry_primary_interval=600


# Interim accounting update interval
# If this is set (larger than 0) and acct_server is configured, hostapd will
# send interim accounting updates every N seconds. Note: if set, this overrides
# possible Acct-Interim-Interval attribute in Access-Accept message. Thus, this
# value should not be configured in hostapd.conf, if RADIUS server is used to
# control the interim interval.
# This value should not be less 600 (10 minutes) and must not be less than
# 60 (1 minute).
#radius_acct_interim_interval=600

# Dynamic VLAN mode; allow RADIUS authentication server to decide which VLAN
# is used for the stations. This information is parsed from following RADIUS
# attributes based on RFC 3580 and RFC 2868: Tunnel-Type (value 13 = VLAN),
# Tunnel-Medium-Type (value 6 = IEEE 802), Tunnel-Private-Group-ID (value
# VLANID as a string). vlan_file option below must be configured if dynamic
# VLANs are used. Optionally, the local MAC ACL list (accept_mac_file) can be
# used to set static client MAC address to VLAN ID mapping.
# 0 = disabled (default)
# 1 = option; use default interface if RADIUS server does not include VLAN ID
# 2 = required; reject authentication if RADIUS server does not include VLAN ID
#dynamic_vlan=0

# VLAN interface list for dynamic VLAN mode is read from a separate text file.
# This list is used to map VLAN ID from the RADIUS server to a network
# interface. Each station is bound to one interface in the same way as with
# multiple BSSIDs or SSIDs. Each line in this text file is defining a new
# interface and the line must include VLAN ID and interface name separated by
# white space (space or tab).
```

```
#vlan_file=/etc/hostapd.vlan

# Interface where 802.1q tagged packets should appear when a RADIUS server is
# used to determine which VLAN a station is on.  hostapd creates a bridge for
# each VLAN.  Then hostapd adds a VLAN interface (associated with the interface
# indicated by 'vlan_tagged_interface') and the appropriate wireless interface
# to the bridge.
#vlan_tagged_interface=eth0


##### RADIUS authentication server configuration ##############################

# hostapd can be used as a RADIUS authentication server for other hosts. This
# requires that the integrated EAP server is also enabled and both
# authentication services are sharing the same configuration.

# File name of the RADIUS clients configuration for the RADIUS server. If this
# commented out, RADIUS server is disabled.
#radius_server_clients=/etc/hostapd.radius_clients

# The UDP port number for the RADIUS authentication server
#radius_server_auth_port=1812

# Use IPv6 with RADIUS server (IPv4 will also be supported using IPv6 API)
#radius_server_ipv6=1


##### WPA/IEEE 802.11i configuration ##########################################

# Enable WPA. Setting this variable configures the AP to require WPA (either
# WPA-PSK or WPA-RADIUS/EAP based on other configuration). For WPA-PSK, either
# wpa_psk or wpa_passphrase must be set and wpa_key_mgmt must include WPA-PSK.
# For WPA-RADIUS/EAP, ieee8021x must be set (but without dynamic WEP keys),
# RADIUS authentication server must be configured, and WPA-EAP must be included
# in wpa_key_mgmt.
# This field is a bit field that can be used to enable WPA (IEEE 802.11i/D3.0)
# and/or WPA2 (full IEEE 802.11i/RSN):
# bit0 = WPA
# bit1 = IEEE 802.11i/RSN (WPA2) (dot11RSNAEnabled)
# 7920 doesn't support WPA2
wpa=1

# WPA pre-shared keys for WPA-PSK. This can be either entered as a 256-bit
# secret in hex format (64 hex digits), wpa_psk, or as an ASCII passphrase
# (8..63 characters) that will be converted to PSK. This conversion uses SSID
# so the PSK changes when ASCII passphrase is used and the SSID is changed.
# wpa_psk (dot11RSNAConfigPSKValue)
# wpa_passphrase (dot11RSNAConfigPSKPassPhrase)
#wpa_psk=0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef
wpa_passphrase=example-password

# Optionally, WPA PSKs can be read from a separate text file (containing list
# of (PSK,MAC address) pairs. This allows more than one PSK to be configured.
# Use absolute path name to make sure that the files can be read on SIGHUP
# configuration reloads.
#wpa_psk_file=/etc/hostapd.wpa_psk

# Set of accepted key management algorithms (WPA-PSK, WPA-EAP, or both). The
# entries are separated with a space. WPA-PSK-SHA256 and WPA-EAP-SHA256 can be
# added to enable SHA256-based stronger algorithms.
# (dot11RSNAConfigAuthenticationSuitesTable)
wpa_key_mgmt=WPA-PSK
```

```
# Set of accepted cipher suites (encryption algorithms) for pairwise keys
# (unicast packets). This is a space separated list of algorithms:
# CCMP = AES in Counter mode with CBC-MAC [RFC 3610, IEEE 802.11i/D7.0]
# TKIP = Temporal Key Integrity Protocol [IEEE 802.11i/D7.0]
# Group cipher suite (encryption algorithm for broadcast and multicast frames)
# is automatically selected based on this configuration. If only CCMP is
# allowed as the pairwise cipher, group cipher will also be CCMP. Otherwise,
# TKIP will be used as the group cipher.
# (dot11RSNAConfigPairwiseCiphersTable)
# Pairwise cipher for WPA (v1) (default: TKIP)
# 7920 only supports TKIP
wpa_pairwise=TKIP
# Pairwise cipher for RSN/WPA2 (default: use wpa_pairwise value)
#rsn_pairwise=CCMP

# Time interval for rekeying GTK (broadcast/multicast encryption keys) in
# seconds. (dot11RSNAConfigGroupRekeyTime)
#wpa_group_rekey=600

# Rekey GTK when any STA that possesses the current GTK is leaving the BSS.
# (dot11RSNAConfigGroupRekeyStrict)
#wpa_strict_rekey=1

# Time interval for rekeying GMK (master key used internally to generate GTKs
# (in seconds).
#wpa_gmk_rekey=86400

# Maximum lifetime for PTK in seconds. This can be used to enforce rekeying of
# PTK to mitigate some attacks against TKIP deficiencies.
#wpa_ptk_rekey=600

# Enable IEEE 802.11i/RSN/WPA2 pre-authentication. This is used to speed up
# roaming be pre-authenticating IEEE 802.1X/EAP part of the full RSN
# authentication and key handshake before actually associating with a new AP.
# (dot11RSNAPreauthenticationEnabled)
#rsn_preauth=1
#
# Space separated list of interfaces from which pre-authentication frames are
# accepted (e.g., 'eth0' or 'eth0 wlan0wds0'. This list should include all
# interface that are used for connections to other APs. This could include
# wired interfaces and WDS links. The normal wireless data interface towards
# associated stations (e.g., wlan0) should not be added, since
# pre-authentication is only used with APs other than the currently associated
# one.
#rsn_preauth_interfaces=eth0

# peerkey: Whether PeerKey negotiation for direct links (IEEE 802.11e) is
# allowed. This is only used with RSN/WPA2.
# 0 = disabled (default)
# 1 = enabled
#peerkey=1

# ieee80211w: Whether management frame protection (MFP) is enabled
# 0 = disabled (default)
# 1 = optional
# 2 = required
#ieee80211w=0

# Association SA Query maximum timeout (in TU = 1.024 ms; for MFP)
# (maximum time to wait for a SA Query response)
# dot11AssociationSAQueryMaximumTimeout, 1...4294967295
#assoc_sa_query_max_timeout=1000
```

```
# Association SA Query retry timeout (in TU = 1.024 ms; for MFP)
# (time between two subsequent SA Query requests)
# dot11AssociationSAQueryRetryTimeout, 1...4294967295
#assoc_sa_query_retry_timeout=201

# disable_pmksa_caching: Disable PMKSA caching
# This parameter can be used to disable caching of PMKSA created through EAP
# authentication. RSN preauthentication may still end up using PMKSA caching if
# it is enabled (rsn_preauth=1).
# 0 = PMKSA caching enabled (default)
# 1 = PMKSA caching disabled
#disable_pmksa_caching=0

# okc: Opportunistic Key Caching (aka Proactive Key Caching)
# Allow PMK cache to be shared opportunistically among configured interfaces
# and BSSes (i.e., all configurations within a single hostapd process).
# 0 = disabled (default)
# 1 = enabled
#okc=1


##### IEEE 802.11r configuration ##############################################

# Mobility Domain identifier (dot11FTMobilityDomainID, MDID)
# MDID is used to indicate a group of APs (within an ESS, i.e., sharing the
# same SSID) between which a STA can use Fast BSS Transition.
# 2-octet identifier as a hex string.
#mobility_domain=a1b2

# PMK-R0 Key Holder identifier (dot11FTR0KeyHolderID)
# 1 to 48 octet identifier.
# This is configured with nas_identifier (see RADIUS client section above).

# Default lifetime of the PMK-RO in minutes; range 1..65535
# (dot11FTR0KeyLifetime)
#r0_key_lifetime=10000

# PMK-R1 Key Holder identifier (dot11FTR1KeyHolderID)
# 6-octet identifier as a hex string.
#r1_key_holder=000102030405

# Reassociation deadline in time units (TUs / 1.024 ms; range 1000..65535)
# (dot11FTReassociationDeadline)
#reassociation_deadline=1000

# List of R0KHs in the same Mobility Domain
# format: <MAC address> <NAS Identifier> <128-bit key as hex string>
# This list is used to map R0KH-ID (NAS Identifier) to a destination MAC
# address when requesting PMK-R1 key from the R0KH that the STA used during the
# Initial Mobility Domain Association.
#r0kh=02:01:02:03:04:05 r0kh-1.example.com 000102030405060708090a0b0c0d0e0f
#r0kh=02:01:02:03:04:06 r0kh-2.example.com 00112233445566778899aabbccddeeff
# And so on.. One line per R0KH.

# List of R1KHs in the same Mobility Domain
# format: <MAC address> <R1KH-ID> <128-bit key as hex string>
# This list is used to map R1KH-ID to a destination MAC address when sending
# PMK-R1 key from the R0KH. This is also the list of authorized R1KHs in the MD
# that can request PMK-R1 keys.
#r1kh=02:01:02:03:04:05 02:11:22:33:44:55 000102030405060708090a0b0c0d0e0f
#r1kh=02:01:02:03:04:06 02:11:22:33:44:66 00112233445566778899aabbccddeeff
# And so on.. One line per R1KH.
```

```
# Whether PMK-R1 push is enabled at R0KH
# 0 = do not push PMK-R1 to all configured R1KHs (default)
# 1 = push PMK-R1 to all configured R1KHs whenever a new PMK-R0 is derived
#pmk_r1_push=1

##### Neighbor table ##########################################################
# Maximum number of entries kept in AP table (either for neigbor table or for
# detecting Overlapping Legacy BSS Condition). The oldest entry will be
# removed when adding a new entry that would make the list grow over this
# limit. Note! WFA certification for IEEE 802.11g requires that OLBC is
# enabled, so this field should not be set to 0 when using IEEE 802.11g.
# default: 255
#ap_table_max_size=255

# Number of seconds of no frames received after which entries may be deleted
# from the AP table. Since passive scanning is not usually performed frequently
# this should not be set to very small value. In addition, there is no
# guarantee that every scan cycle will receive beacon frames from the
# neighboring APs.
# default: 60
#ap_table_expiration_time=3600


##### Wi-Fi Protected Setup (WPS) #############################################

# WPS state
# 0 = WPS disabled (default)
# 1 = WPS enabled, not configured
# 2 = WPS enabled, configured
#wps_state=2

# AP can be configured into a locked state where new WPS Registrar are not
# accepted, but previously authorized Registrars (including the internal one)
# can continue to add new Enrollees.
#ap_setup_locked=1

# Universally Unique IDentifier (UUID; see RFC 4122) of the device
# This value is used as the UUID for the internal WPS Registrar. If the AP
# is also using UPnP, this value should be set to the device's UPnP UUID.
# If not configured, UUID will be generated based on the local MAC address.
#uuid=12345678-9abc-def0-1234-56789abcdef0

# Note: If wpa_psk_file is set, WPS is used to generate random, per-device PSKs
# that will be appended to the wpa_psk_file. If wpa_psk_file is not set, the
# default PSK (wpa_psk/wpa_passphrase) will be delivered to Enrollees. Use of
# per-device PSKs is recommended as the more secure option (i.e., make sure to
# set wpa_psk_file when using WPS with WPA-PSK).

# When an Enrollee requests access to the network with PIN method, the Enrollee
# PIN will need to be entered for the Registrar. PIN request notifications are
# sent to hostapd ctrl_iface monitor. In addition, they can be written to a
# text file that could be used, e.g., to populate the AP administration UI with
# pending PIN requests. If the following variable is set, the PIN requests will
# be written to the configured file.
#wps_pin_requests=/var/run/hostapd_wps_pin_requests

# Device Name
# User-friendly description of device; up to 32 octets encoded in UTF-8
#device_name=Wireless AP

# Manufacturer
# The manufacturer of the device (up to 64 ASCII characters)
#manufacturer=Company
```

```
# Model Name
# Model of the device (up to 32 ASCII characters)
#model_name=WAP

# Model Number
# Additional device description (up to 32 ASCII characters)
#model_number=123

# Serial Number
# Serial number of the device (up to 32 characters)
#serial_number=12345

# Primary Device Type
# Used format: <categ>-<OUI>-<subcateg>
# categ = Category as an integer value
# OUI = OUI and type octet as a 4-octet hex-encoded value; 0050F204 for
#       default WPS OUI
# subcateg = OUI-specific Sub Category as an integer value
# Examples:
#   1-0050F204-1 (Computer / PC)
#   1-0050F204-2 (Computer / Server)
#   5-0050F204-1 (Storage / NAS)
#   6-0050F204-1 (Network Infrastructure / AP)
#device_type=6-0050F204-1

# OS Version
# 4-octet operating system version number (hex string)
#os_version=01020300

# Config Methods
# List of the supported configuration methods
# Available methods: usba ethernet label display ext_nfc_token int_nfc_token
# nfc_interface push_button keypad virtual_display physical_display
# virtual_push_button physical_push_button
#config_methods=label virtual_display virtual_push_button keypad

# WPS capability discovery workaround for PBC with Windows 7
# Windows 7 uses incorrect way of figuring out AP's WPS capabilities by acting
# as a Registrar and using M1 from the AP. The config methods attribute in that
# message is supposed to indicate only the configuration method supported by
# the AP in Enrollee role, i.e., to add an external Registrar. For that case,
# PBC shall not be used and as such, the PushButton config method is removed
# from M1 by default. If pbc_in_m1=1 is included in the configuration file,
# the PushButton config method is left in M1 (if included in config_methods
# parameter) to allow Windows 7 to use PBC instead of PIN (e.g., from a label
# in the AP).
#pbc_in_m1=1

# Static access point PIN for initial configuration and adding Registrars
# If not set, hostapd will not allow external WPS Registrars to control the
# access point. The AP PIN can also be set at runtime with hostapd_cli
# wps_ap_pin command. Use of temporary (enabled by user action) and random
# AP PIN is much more secure than configuring a static AP PIN here. As such,
# use of the ap_pin parameter is not recommended if the AP device has means for
# displaying a random PIN.
#ap_pin=12345670

# Skip building of automatic WPS credential
# This can be used to allow the automatically generated Credential attribute to
# be replaced with pre-configured Credential(s).
#skip_cred_build=1
```

```
# Additional Credential attribute(s)
# This option can be used to add pre-configured Credential attributes into M8
# message when acting as a Registrar. If skip_cred_build=1, this data will also
# be able to override the Credential attribute that would have otherwise been
# automatically generated based on network configuration. This configuration
# option points to an external file that much contain the WPS Credential
# attribute(s) as binary data.
#extra_cred=hostapd.cred

# Credential processing
#   0 = process received credentials internally (default)
#   1 = do not process received credentials; just pass them over ctrl_iface to
# external program(s)
#   2 = process received credentials internally and pass them over ctrl_iface
# to external program(s)
# Note: With wps_cred_processing=1, skip_cred_build should be set to 1 and
# extra_cred be used to provide the Credential data for Enrollees.
#
# wps_cred_processing=1 will disabled automatic updates of hostapd.conf file
# both for Credential processing and for marking AP Setup Locked based on
# validation failures of AP PIN. An external program is responsible on updating
# the configuration appropriately in this case.
#wps_cred_processing=0

# AP Settings Attributes for M7
# By default, hostapd generates the AP Settings Attributes for M7 based on the
# current configuration. It is possible to override this by providing a file
# with pre-configured attributes. This is similar to extra_cred file format,
# but the AP Settings attributes are not encapsulated in a Credential
# attribute.
#ap_settings=hostapd.ap_settings

# WPS UPnP interface
# If set, support for external Registrars is enabled.
#upnp_iface=br0

# Friendly Name (required for UPnP)
# Short description for end use. Should be less than 64 characters.
#friendly_name=WPS Access Point

# Manufacturer URL (optional for UPnP)
#manufacturer_url=http://www.example.com/

# Model Description (recommended for UPnP)
# Long description for end user. Should be less than 128 characters.
#model_description=Wireless Access Point

# Model URL (optional for UPnP)
#model_url=http://www.example.com/model/

# Universal Product Code (optional for UPnP)
# 12-digit, all-numeric code that identifies the consumer package.
#upc=123456789012

##### Wi-Fi Direct (P2P) ######################################################

# Enable P2P Device management
#manage_p2p=1

# Allow cross connection
#allow_cross_connection=1

#### TDLS (IEEE 802.11z-2010) #################################################
```

```
# Prohibit use of TDLS in this BSS
#tdls_prohibit=1

# Prohibit use of TDLS Channel Switching in this BSS
#tdls_prohibit_chan_switch=1

##### IEEE 802.11v-2011 #######################################################

# Time advertisement
# 0 = disabled (default)
# 2 = UTC time at which the TSF timer is 0
#time_advertisement=2

# Local time zone as specified in 8.3 of IEEE Std 1003.1-2004:
# stdoffset[dst[offset][,start[/time],end[/time]]]
#time_zone=EST5

##### IEEE 802.11u-2011 #######################################################

# Enable Interworking service
#interworking=1

# Access Network Type
# 0 = Private network
# 1 = Private network with guest access
# 2 = Chargeable public network
# 3 = Free public network
# 4 = Personal device network
# 5 = Emergency services only network
# 14 = Test or experimental
# 15 = Wildcard
#access_network_type=0

# Whether the network provides connectivity to the Internet
# 0 = Unspecified
# 1 = Network provides connectivity to the Internet
#internet=1

# Additional Step Required for Access
# Note: This is only used with open network, i.e., ASRA shall ne set to 0 if
# RSN is used.
#asra=0

# Emergency services reachable
#esr=0

# Unauthenticated emergency service accessible
#uesa=0

# Venue Info (optional)
# The available values are defined in IEEE Std 802.11u-2011, 7.3.1.34.
# Example values (group,type):
# 0,0 = Unspecified
# 1,7 = Convention Center
# 1,13 = Coffee Shop
# 2,0 = Unspecified Business
# 7,1  Private Residence
#venue_group=7
#venue_type=1

# Homogeneous ESS identifier (optional; dot11HESSID)
# If set, this shall be identifical to one of the BSSIDs in the homogeneous
```

```
# ESS and this shall be set to the same value across all BSSs in homogeneous
# ESS.
#hessid=02:03:04:05:06:07

# Roaming Consortium List
# Arbitrary number of Roaming Consortium OIs can be configured with each line
# adding a new OI to the list. The first three entries are available through
# Beacon and Probe Response frames. Any additional entry will be available only
# through ANQP queries. Each OI is between 3 and 15 octets and is configured a
# a hexstring.
#roaming_consortium=021122
#roaming_consortium=2233445566

##### Multiple BSSID support ##################################################
#
# Above configuration is using the default interface (wlan#, or multi-SSID VLAN
# interfaces). Other BSSIDs can be added by using separator 'bss' with
# default interface name to be allocated for the data packets of the new BSS.
#
# hostapd will generate BSSID mask based on the BSSIDs that are
# configured. hostapd will verify that dev_addr & MASK == dev_addr. If this is
# not the case, the MAC address of the radio must be changed before starting
# hostapd (ifconfig wlan0 hw ether <MAC addr>). If a BSSID is configured for
# every secondary BSS, this limitation is not applied at hostapd and other
# masks may be used if the driver supports them (e.g., swap the locally
# administered bit)
#
# BSSIDs are assigned in order to each BSS, unless an explicit BSSID is
# specified using the 'bssid' parameter.
# If an explicit BSSID is specified, it must be chosen such that it:
# - results in a valid MASK that covers it and the dev_addr
# - is not the same as the MAC address of the radio
# - is not the same as any other explicitly specified BSSID
#
# Please note that hostapd uses some of the values configured for the first BSS
# as the defaults for the following BSSes. However, it is recommended that all
# BSSes include explicit configuration of all relevant configuration items.
#
#bss=wlan0_0
#ssid=test2
# most of the above items can be used here (apart from radio interface specific
# items, like channel)

#bss=wlan0_1
#bssid=00:13:10:95:fe:0b
# ...
```

5. Update the following parameters (if applicable) in the configuration file:

   - interface
   - ssid
   - channel
   - wpa_passphrase

6. Create a new stanza in `/etc/network/interfaces`:

```
iface wlan-sccp inet manual
     hostapd /etc/hostapd/hostapd.sccp.conf
```

7. Up the interface:

```
ifup wlan0=wlan-sccp
```

8. Configure your 7920/7921 to connect to the network.

   To unlock the phone's configuration menu on the 7921:

   - Press the Navigation Button downwards to enter SETTINGS mode
   - Navigate to and select Network Profiles
   - Unlock the IP phone's configuration menu by pressing **#. The padlock icon on the top-right of the screen will change from closed to open.

   When asked for the authentication mode, select something like "Auto" or "AKM".

   You don't have to enter anything for the username/password.

9. You'll probably want to bridge your wlan0 interface with another interface, for example a VLAN interface:

   ```
   brctl addbr br0
   brctl addif br0 wlan0
   brctl addif br0 eth0.341
   ip link set br0 up
   ```

10. If you are using virtualbox and your guest interface is bridged to eth0.341, you'll need to change its configuration and bridge it with br0 instead, else it won't work properly.

1.12.13 Web Interface

Configuration for development

Default error level for XiVO web interface is E_ALL & ~E_DEPRECATED & ~E_USER_DEPRECATED & ~E_RECOVERABLE_ERROR & ~E_STRICT

If you want to display warning or other error in your browser, edit the `/etc/xivo/web-interface/xivo.ini` and replace report_type level to 3:

```
[error]
level = 2047
report_type = 3
report_mode = 1
report_func = 1
email = john.doe@example.com
file = /var/log/xivo-web-interface/error.log
```

You may also edit `/etc/xivo/web-interface/php.ini` and change the error level, but you will need to restart the cgi:

```
/etc/init.d/spawn-fcgi restart
```

Interactive debugging in Eclipse

On your XiVO:

1. Install php5-xdebug:

   ```
   $ apt-get install php5-xdebug
   ```

2. Edit the `/etc/php5/conf.d/xdebug.ini` and add these lines at the end:

   ```
   xdebug.remote_enable=On
   xdebug.remote_host="<dev_host_ip>"
   xdebug.remote_port=9000
   xdebug.remote_handler="dbgp"
   ```

```
where ``<dev_host_ip>`` is the IP address of your machine where Eclipse is installed.
Of course, your XiVO must be able to reach this IP address.
```

3. Restart spawn-fcgi:

```
$ /etc/init.d/spawn-fcgi restart
```

On your machine where Eclipse is installed:

1. Make sure you have Eclipse PDT installed
2. In the Eclipse preferences, on the PHP / Debug page:
   - Set the PHP Debugger to XDebug
   - Add a new PHP server with the following information:
     - Name: anything you want
     - Base URL: `https://<xivo_ip>`
3. Create a new `PHP Web Application` debug configuration:
   - Choose the PHP server you create on last step
   - Pick some file, which can be anything if you don't "break at first line"
   - Uncheck "Auto Generate", and set the path you want your browser to open when you'll launch this debug configuration.

Then, to start a debugging session, set some breakpoints in the code and launch your debug configuration. This will open the page in your browser, and when the code will hit your breakpoints, you'll be able to go through the code step by step, etc.

1.12.14 XiVO Client

Building the XiVO Client

Building the XiVO Client on Windows platforms

This page explains how to build an executable of the XiVO Client from its sources for Windows.

Windows Prerequisites

Cygwin Cygwin Web site

Click the "setup" link and execute.

During the installer, check the package:

- Devel > git

Qt SDK You need the development files of the Qt 5 library, available on the Qt website.

NSIS (optional) You will only need NSIS installed if you want to create an installer for the XiVO Client.

NSIS download page

During the installer, choose the full installation.

Get sources In a **Cygwin shell**:

```
git clone git://github.com/xivo-pbx/xivo-client-qt.git
cd xivo-client-qt
```

Building

Path configuration You must change the values in `C:\Cygwin\home\user\xivo-client-qt\build-deps` to match the paths of your installed programs. You must use an editor capable of understanding Unix end of lines, such as Notepad++.

Replace `C:\` with `/cygdrive/c` and backslashes (`\`) with slashes (`/`). You must respect the case of the directory names. Paths containing spaces must be enclosed in double quotes (`"`).

For example, if you installed NSIS in `C:\Program Files (x86)\nsis`, you should write:

```
WIN_NSIS_PATH="/cygdrive/c/Program files (x86)/nsis"
```

Build In a **Cygwin shell**:

```
source build-deps
export PATH=$WIN_QT_PATH/bin:$WIN_MINGW_PATH/bin:$PATH

qmake
mingw32-make
```

Binaries are available in the `bin` directory.

The version of the executable is taken from the `git describe` command.

Launch You can launch the built executable with:

```
source build_deps
PATH=$WIN_QT_PATH/bin:$PATH bin/xivoclient
```

Package To create the installer:

```
mingw32-make pack
```

This will result in a `.exe` file in the current directory.

Build options To add a console:

```
qmake CONFIG+=console
```

To generate debug symbols:

```
mingw32-make DEBUG=yes
```

Clean

```
mingw32-make distclean
```

Building the XiVO Client on GNU/Linux platforms

This page explains how to build an executable of the XiVO Client from its sources for GNU/Linux.

Prerequisites

- Qt5 library development files: Qt website
- Git (package `git`)
- Generic software building tools : `make`, `g++` ... (package `build-essential`)

Get sources In a bash shell:

```
$ git clone git://github.com/xivo-pbx/xivo-client-qt.git
```

Building You need to have the Qt5 binaries (qmake, lrelease, ...) in your $PATH.

Launch qmake to generate the Makefile:

```
$ cd xivo-client-qt
$ /path/to/qt5/bin/qmake
```

This will also generate a file `versions.mak` that contains version informations about the code being compiled. It is necessary for compilation and packaging.

You can then launch `make`:

```
$ make
```

Binaries are available in the `bin` directory.

The version of the executable is taken from the `git describe` command.

Build options To generate debug symbols:

```
$ make DEBUG=yes
```

To compile the unit tests of the XiVO Client:

```
$ qmake CONFIG+=tests
```

or, if you have a recent version of Google Mock (e.g. on Debian Wheezy):

```
$ qmake CONFIG+=tests CONFIG+=gmock
```

To compile the XiVO Client ready for functional tests:

```
$ make FUNCTESTS=yes
```

Cleaning

```
$ make distclean
```

Launch You can launch the built executable with:

```
$ LD_LIBRARY_PATH=bin bin/xivoclient
```

Package To create the Debian package, usable on Debian and Ubuntu, you first need to modify `build-deps` to locate the Qt 5 installation directory:

```
$ /path/to/qt5/bin/qmake -spec linux-g++
$ make
$ make pack
```

This will result in a `.deb` file in the current directory.

The version of the package is taken from the `git describe` command.

Building the XiVO Client on Mac OS

This page explains how to build an executable of the XiVO Client from its sources for Mac OS.

Mac OS Prerequisites

Developer tools You will need an Apple developer account to get development tools, such as GCC. To log in or sign in, go to http://connect.apple.com. In the Downloads section, get the Command line Tools for XCode and install them. You might want to get XCode too, but it is rather big.

Qt SDK You need the development files of the Qt 5 library, available on the Qt website.

Get sources In a bash shell, enter:

```
$ git clone git://github.com/xivo-pbx/xivo-client-qt.git
```

Building Launch qmake to generate the Makefile:

```
$ cd xivo-client-qt
$ /path/to/qt5/bin/qmake -spec macx-g++
```

This will also generate a file `versions.mak` that contains version informations about the code being compiled. It is necessary for compilation and packaging.

You can then launch `make`:

```
$ make
```

Binaries are available in the `bin` directory.

The version of the executable is taken from the `git describe` command.

Debug build Add `DEBUG=yes` on the command line:

```
$ make DEBUG=yes
```

Cleaning

```
$ make distclean
```

Launch You can launch the built executable with:

```
$ DYLD_LIBRARY_PATH=bin bin/xivoclient.app/Contents/MacOS/xivoclient
```

Package You need to have the bin directory of Qt in your $PATH.

To create the app bundle:

```
$ make pack
```

This will result in a `.dmg` file in the current directory.

The version of the package is taken from the `git describe` command.

Coding the XiVO Client

Project folder map

baselib The folder *baselib* contains all files necessary to build the baselib. It contains the necessary code and data structures to communicate with the XiVO CTI server.

This library is designed to be reusable by other XiVO CTI clients. If you want to build it without the rest of the XiVO Client, go in its folder and type:

```
$ qmake && make
```

The library will be available in the new bin folder.

xivoclient The folder *xivoclient* contains all other source files included in the XiVO Client.

src contains the source code files, *images* contains the images, *i18n* contains the translation files and *qtaddons* contains some Qt addons used by the XiVO Client.

src The source files are separated in three categories :

- the XiVO Client itself, the source files are directly in *src*.
- the XLet library (*xletlib*) contains the code common to multiple XLets (plugins), like the XLet base class and mainly GUI stuff.
- the XLets themselves (*xlets*), each one is in a *xlets/something* subfolder.

Each XLet is compiled into a dynamic library, but some XLets are still compiled within the xivoclient executable instead of in a separated library. They are marked with a **-builtin* subfolder name.

delivery This folder contains all license informations necessary for the XiVO Client to be redistributed, i.e. the GNU GPLv3 and the additional requirements.

Configuration access

The settings of the application are stored in BaseEngine for runtime and in files when the client is closed :

- *~/.config/XiVO* on GNU/Linux systems
- (what about other platforms?)

There are now 3 sets of functions from BaseEngine that you can use to read/store settings :

getConfig() / setConfig() They are proxy methods to use the BaseConfig object inside BaseEngine. They use QVariantMap to store the settings values. They are currently used to store/retrieve options used in the ConfigWidget.

You can find the available keys to access data in the detailed Doxygen documentation of BaseEngine, or in *baseengine.h*.

Note that the settings stored in BaseConfig won't be written in the configuration file if BaseEngine is not aware of their existence (loaded in *loadSettings* and saved in *saveSettings*).

getSettings() Through this function, you can access the lowest level of configuration storage, QSettings. It also contains the options stored in BaseConfig, but is less easy to use.

This direct access is used for purely graphical settings, only used to remember the appearance of the GUI until the next launch. These settings don't have to be shared with other widgets, and storing them directly in QSettings avoids writing code to import/export to/from BaseConfig.

getProfileSetting() / setProfileSetting() This pair of methods allow you to read/write settings directly in QSettings, but specifically for the current configuration profile.

Configuration profiles

When starting XiVO Client with an argument, this argument is interpreted as a profile name. This profile name allows you to separate different profiles, with different configuration options.

For example, configuration profile "profileA" will auto-connect with user A and password B and "profileB" will not auto-connect, but is set to connect with user C, no password remembered. To invoke these profiles, use :

```
$ xivoclient profileA
$ xivoclient profileB
```

The default configuration profile is default-user.

Recognizing / extracting phone numbers

Of course, working on XiVO Client implies working with phone numbers. But how to interpret them easily, when we are not sure of the format they're in?

You can use the PhoneNumber namespace (*baselib/src/phonenumber.h*) to do that, it contains routines for recognition/extraction of phone numbers, that way you don't have to parse manually.

These subroutines are pretty basic for the moment, if you need/want to improve them, feel free to do it.

Retrieving CTI server infos

Informations are synchronized from the server to the BaseEngine when the client connects.

It is stored in BaseEngine in "lists". It is stored in a format close to the one used to transmit it, so you can see the CTI protocol definition for further documentation.

Each list contains objects of different type. These types are :

- channel
- user
- phone
- trunk
- agent
- queue
- group
- meetme
- voicemail
- queuemember
- parking

Each type corresponds to a class derived from XInfo, e.g. channel infos are stored in ChannelInfo objects.

The basic attributes of all objects are 3 strings: the IPBX ID, the XiVO object ID and the extended ID of the object, which is the two previous attributes linked with a "/".

Listen to IPBX events If you want your XLet to receive IPBX/CTI events, you can do so by inheriting the IPBXListener interface.

You must specify which type of events you want to listen. This depends of the implemented functions in the CTI server. You can register to listen these events by calling the IPBXListener method :

> registerListener(xxx);

For now, *xxx*, the event type, can take take the values : * chitchat * history * records_campaign * queuestats

On reception of the specified type of event, BaseEngine will call the *IPBXListener* method *parseCommand(QVariantMap)*.

You should then reimplement this method to make it process the event data, stored in the *QVariantMap* parameter.

The parking XLet

There are two concepts here : * Parked calls: These calls have been parked by a switchboard or an operator. They are waiting to be answered by a specific person, unlike a queue, where calls will be answered by one of the agents of the group associated to the queue. Each parked call is given a phone number so that the call can be answered by everyone.

- Parking lots: They are containers for parked calls. Each parking lot has a phone number, used to identify where to send the call we want to park.

ParkingWidget represents a parking lot and contains a table that stores all parked calls.

Adding new XLets

When you want to add a new XLet, you can use the basic XLetNull, that only prints "Hello World". Here is a little script to accelerate the copy from XLetNull.

```
#!/usr/bin/env sh

newname="newname" # Replaces xletnull
NewName="NewName" # Replaces XLetNull & XletNull
NEWNAME="NEWNAME" # Replaces XLETNULL

if [ ! -d xletnull ] ; then

    echo "Please execute this script in XIVO_CLIENT/plugins"
    echo $newname
    exit 1
fi

cp -r xletnull $newname
cd $newname
rm -f moc* *.o Makefile

for f in $(find . -type f -print) ; do
    mv $f `echo $f | sed s/xletnull/$newname/`
done

find . -type f -exec sed -i "s/xletnull/$newname/g;s/X[Ll]etNull/$NewName/g;s/XLETNULL/$NEWNAME/g
```

Before executing the script, just replace the first three variables with the name of the new XLet.

Then, you must add a line in xivoclient/xlets.pro to add your new directory to the SUBDIRS variable.

Then you can start implementing your new class. The <xletname>Plugin class is only an interface between the main app and your XLet.

Translations If you want to localize your XLet, there are four steps.

Modify the sources In the <xletname>Plugin constructor, add the line :

b_engine->registerTranslation(":/<xletname>_%1");

before the return instruction.

Modify the project file Add these lines in the .pro file in your XLet directory :

TRANSLATIONS = <xletname>_fr.ts TRANSLATIONS += <xletname>_nl.ts

RESOURCES = res.qrc

Replace fr and nl with the languages you want.

Create the resource file In a file res.qrc in your XLet directory, put these lines :

```
<!DOCTYPE RCC><RCC version="1.0">
    <qresource>
        <file><xletname>_fr.qm</file>
        <file><xletname>_nl.qm</file>
    </qresource>
</RCC>
```

These files will be embedded in the Xlet library binary.

Create the translation files In your XLet directory, run :

lupdate <xletname>.pro

This creates as much .ts translation files as specified in the .pro file. You can now translate strings in these file.

The XLet will now be compiled and translated.

Add a new XLet

For now, it is not possible to add easily an XLet without changing the CTI server configuration files.

If you just want to test your new XLet, you can add the following line in baseengine.cpp :

m_capaxlets.push_back(QVariantList() << QVariant("<xletname>") << QVariant("tab"));

right after the line

m_capaxlets = datamap.value("capaxlets").toList();

You can replace "tab" with "grid" or "dock".

Add a translation

This is definitely not something funny and not easy to automatize.

You have to add, in every .pro file of the project (except xlets.pro and all those that don't need translations), a line

TRANSLATIONS += <project>_<lang>.ts

Replace <project> with the project name (xivoclient, baselib, xlet) and <lang> by the identifier of your language (en, fr, nl, ...) Then you have to add, in every .qrc file, the .qm files corresponding to the ones you added in the .pro files, such as :

<file><project>_<lang>.qm</file>

in the <qresource> section of these XML .qrc files.

After that, you have to run, in the XiVO Client root directory, something like :

> find . -name *.pro -exec lupdate {} ;

This will create or update all .ts translation files registered in the .pro files.

You can then start translating the strings in these files, in the *xivoclient/i18n* folder.

Code modification

If you want to be able to select your new language from within the XiVO Client, you have to add it in the interface.

For that, you can add your new language in the *m_locale_cbox* QCombobox in ConfigWidget.

CTI debugging tool

If you have a problem and you want to see what is going on between the CTI server and client, you can use a specific script, designed specifically for XiVO, instead of using something like Wireshark to listen network communications.

Profiling

To get profiling informations on the XiVO Client:

- Compile the XiVO Client with debugging symbols
- Run the command:

  ```
  LD_LIBRARY_PATH=bin valgrind --tool=callgrind bin/xivoclient
  ```

- Quit the client
- Open the generated file `callgrind.out.<pid>` with KCacheGrind

Automatic checking tools

We use two tools to check the source code of the XiVO Client: CppCheck et Valgrind.

CppCheck Usage:

```
cppcheck -I baselib/src -I xivoclient/src .
```

Valgrind (Memcheck) Usage:

```
LD_LIBRARY_PATH=bin valgrind --leak-check=full --suppressions=valgrind.supp --num-callers=30 --ge
```

You need to fill a file `valgrind.supp` with Valgrind suppressions, to avoid displaying errors in code you have no control over.

Here is a template `valgrind.supp` you can use. All memory in the XiVO Client is allocated using the new operator, so all calls to `malloc` and co. must come from libraries:

```
{
   malloc
   Memcheck:Leak
   fun:malloc
   ...
}

{
   calloc
```

```
   Memcheck:Leak
   fun:calloc
   ...
}

{
   realloc
   Memcheck:Leak
   fun:realloc
   ...
}

{
   memalign
   Memcheck:Leak
   fun:memalign
   ...
}
```

Figures

Here's a call graph for the presence features. Not complete, but gives a good global view of the internal mechanism.

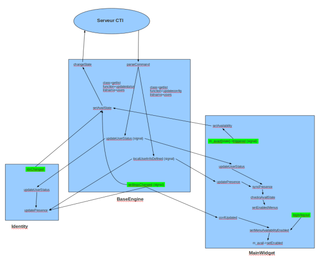


Figure 1.90: Xivo Client presence call graph

Here's a call graph describing the chaining of calls when the XiVO Client connects to the server.

Manage Translations of the XiVO Client

This sections describes how to manage XiVO Client translations from a developer point of view. If you want to help translate the XiVO Client, see *Translating XiVO*

You need to install these tools:

```
pip install transifex-client
apt-get install qt4-dev-tools
```

How to Add a New Translated String

String to be translated is marked using the tr macro in the source code.

Example:

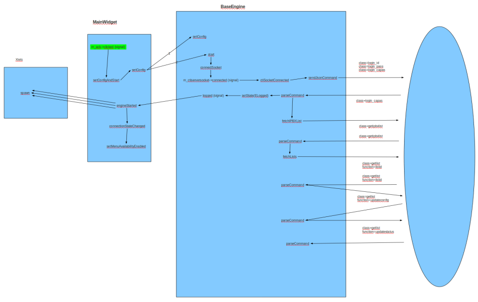


Figure 1.91: Xivo Client login call graph

```
tr("Number");
```

Updating translations on transifex Run the following commands from the root of the xivo-client-qt project:

```
utils/translations.sh pull
utils/translations.sh commit
utils/translations.sh push
```

After these first 3 commands, you can visit transifex, and check that the xivo-client is 100% translated for your language. Once all the translations have been checked, run the 3 following commands:

```
utils/translations.sh pull
git commit
git push
```

> **Warning:** Under Arch Linux, you must have qt4 installed and prepend `PATH=/usr/lib/qt4/bin:$PATH` before each command

Add a new XiVO Client locale

Localizing the XiVO Client goes through four steps :

- Creating the new translation in Transifex
- Generate the translation files
- Embedding the translation in the binaries
- Display the new locale to be chosen

Creating the new translation in Transifex Log into Transifex and click the 'Create language' option.

Generate translation files The translation files will be automatically generated from the source code.

For the command to create files for your locale, you need to ensure it is listed in the project file.

There are a few project files you should edit, each one will translate a module of the XiVO Client :

- `baselib/baselib.pro`
- `xivoclient/xivoclient.pro`
- `xivoclient/xletlib.pro`

- `xivoclient/src/xlets/*/*.pro`

In these files, you should add a line like this one:

```
TRANSLATIONS += $$ROOT_DIR/i18n/xivoclient_fr.ts
```

This line adds a translation file for french. Please replace fr by the code of your locale. The `$$ROOT_DIR` variable references either xivoclient or baselib.

You can use a command like the following to automate this ($LANG is the new language):

```
find . -name '*.pro' -exec sed -i -e 's|^TRANSLATIONS += $${\?ROOT_DIR}\?/i18n/\(.*\)_en.ts|\0\nTI
```

To actually create the files, you will have to use the translation managing script. But first, you must tell the script about your new locale. Edit the `utils/translations.sh` file and add your locale to the `LOCALES` variable. Then, you can run the script:

```
$ utils/translations.sh update
```

Embed the translation files For each project previously edited, you should have a corresponding `.qrc` file. These resource files list all files that will be embedded in the XiVO Client binaries. You should then add the corresponding translation files like below:

```
<file>obj/xivoclient_fr.qm</file>
```

This embeds the French translation of the `xivoclient` module, corresponding to the translation file above. The path is changed to `obj/` because the `.qm` file will be generated from the `.ts` file.

You can use a command like the following to automate this ($LANG is the new language):

```
find . -name '*.qrc' -exec sed -i -e 's|^\( *\)<file>\(.*\)obj/\(.*\)_fr.qm</file>|\0\n\1<file>\2
```

Display the new locale You have to edit the source file `xivoclient/src/configwidget.cpp` and add the entry corresponding to your locale in the locale-choosing combobox.

Checklist:

1.12.15 Adding support for a new SCCP phone

Introduction

This page describes the requirements to considered that a SCCP phone is working with XiVO libsccp.

Checklist

Basic functionnality

- Register on Asterisk
- SCCP reset [restart]
- Call history
- Date time display
- HA

Telephony

These test should be done with and without direct media enabled

- Emit a call
- Receive a call
- Receive and transfer a call
- Emit a call and transfer the call
- Hold and resume a call
- Features (*0 and others)
- Receive 2 calls simultaneously
- Emit 2 calls simultaneously
- DTMF on an external IVR

Function keys

- Redial
- DnD
- Hold
- Resume
- New call
- End call
- Call forward (Enable)
- Call forward (Disable)
- Try each button in each mode (on hook, in progress, etc)

Optionnal options to test and document

- Phone book
- Caller ID and other display i18n
- MWI
- Speeddial/BLF

1.13 Troubleshooting

The list of current bugs can be found on the official XiVO issue tracker.

1.13.1 Transfers using DTMF

When transfering a call using DTMF (*1) you get an *invalid extension* error when dialing the extension.

The workaround to this problem is to create a preprocess subroutine and assign it to the destinations where you have the problem.

Under *Services* → *IPBX* → *IPBX configuration* → *Configuration files* add a new file containing the following dialplan:

```
[allow-transfer]
exten = s,1,NoOp(## Setting transfer context ##)
same = n,Set(__TRANSFER_CONTEXT=<internal-context>)
same = n,Return()
```

Do not forget to substitute <internal-context> with your internal context.

Some places where you might want to add this preprocess subroutine is on queues and outgoing calls to be able to transfer the called person to another extension.

1.13.2 Fax detection

XiVO **does not currently support Fax detection**. The following describe a workaround to use this feature. The behavior is to answer all incoming (external) call, wait for a number of seconds (4 in this example) : if a fax is detected, receive it otherwise route the call normally.

Note: This workaround works only :

- on incoming calls towards an User (and an User only),
- if the incoming trunk is a DAHDI or a SIP trunk,
- if the user has a voicemail which is activated and with the email field filled
- XiVO >= 13.08 (needs asterisk 11)

Be aware that this workaround will probably not survive any upgrade.

1. In the Web Interface and under *Services* → *IPBX* → *IPBX configuration* → *Configuration files* add a new file named *fax-detection.conf* containing the following dialplan:

```
;; Fax Detection
[pre-user-global-faxdetection]
exten = s,1,NoOp(Answer call to be able to detect fax if call is external AND user has an ema
same  =   n,GotoIf($["${XIVO_CALLORIGIN}" = "extern"]?:return)
same  =   n,GotoIf(${XIVO_USEREMAIL}?:return)
same  =   n,Set(FAXOPT(faxdetect)=yes) ; Activate dynamically fax detection
same  =   n,Answer()
same  =   n,Wait(4) ; You can change the number of seconds it will wait for fax (4 to 6 is go
same  =   n,Set(FAXOPT(faxdetect)=no) ; If no fax was detected deactivate dyamically fax dete
same  =   n(return),Return()

exten = fax,1,NoOp(Fax detected from ${CALLERID(num)} towards ${XIVO_DSTNUM} - will be sent u
same  =     n,GotoIf($["${CHANNEL(channeltype)}" = "DAHDI"]?changeechocan:continue)
same  =     n(changeechocan),Set(CHANNEL(echocan_mode)=fax) ; if chan type is dahdi set echo
same  =     n(continue),Gosub(faxtomail,s,1(${XIVO_USEREMAIL}))
```

2. In the file `/etc/xivo/asterisk/xivo_globals.conf` set the global user subroutine to `pre-user-global-faxdetection` : this subroutine will be executed each time a user is called:

```
XIVO_PRESUBR_GLOBAL_USER = pre-user-global-faxdetection
```

3. Reload asterisk configuration (both for dialplan and dahdi):

```
asterisk -rx 'core reload'
```

1.13.3 Berofos Integration with PBX

You can use a Berofos failover switch to secure the ISDN provider lines when installing a XiVO in front of an existing PBX. The goal of this configuration is to mitigate the consequences of an outage of the XiVO : with this

equipment the ISDN provider links could be switched to the PBX directly if the XiVO goes down.

XiVO **does not offer natively** the possibility to configure Berofos in this failover mode. This section describes a workaround.

Logical view:

```
                +------+                               +-----+
-- Provider ----| XiVO | -- ISDN Interconnection  --| PBX | -- Phones
                +------+                               +-----+
```

Connection:

```
  +-------------Bero*fos---------------+
  | A        B        C        D       |
  | o o o o  o o o o  o o o o  o o o o |
  +-+-+------+-+------+-+------+-+-----+
    | |      | |      | |      | |
   / /       | |      | |      | |
  / /    +--------+   / /   +---------+
2 T2     |  XiVO  |  / /    |   PBX   |
         +--------+ / /     +---------+
             | |   / /
             \ \__/ /
              \____/
```

The following describes how to configure your XiVO and your Berofos.

1. Follow the Berofos general configuration (firmware, IP, login/password) described in the the *Berofos Installation and Configuration* page.
2. When done, apply these specific parameters to the berofos:

```
bnfos --set scenario=1   -h 10.105.2.26 -u admin:berofos
bnfos --set mode=1       -h 10.105.2.26 -u admin:berofos
bnfos --set modedef=1    -h 10.105.2.26 -u admin:berofos
bnfos --set wdog=1       -h 10.105.2.26 -u admin:berofos
bnfos --set wdogdef=1    -h 10.105.2.26 -u admin:berofos
bnfos --set wdogitime=60 -h 10.105.2.26 -u admin:berofos
```

3. Add the following script `/usr/local/sbin/berofos-workaround`:

```
#!/bin/bash
# Script workaround for berofos integration with a XiVO in front of PABX

res=$(/etc/init.d/asterisk status)
does_ast_run=$?
if [ $does_ast_run -eq 0 ]; then
    /usr/bin/logger "$0 - Asterisk is running"
    # If asterisk is running, we (re)enable wdog and (re)set the mode
    /usr/bin/bnfos --set mode=1 -f fos1 -s
    /usr/bin/bnfos --set modedef=1 -f fos1 -s
    /usr/bin/bnfos --set wdog=1 -f fos1 -s

    # Now 'kick' berofos ten times each 5 seconds
    for ((i == 1; i <= 10; i += 1)); do
        /usr/bin/bnfos --kick -f fos1 -s
        /bin/sleep 5
    done
else
    /usr/bin/logger "$0 - Asterisk is not running"
fi
```

4. Add execution rights to script:

```
chmod +x /usr/local/sbin/berofos-workaround
```

5. Create a cron to launch the script every minutes `/etc/cron.d/berofos-cron-workaround`:

```
# Workaround to berofos integration
MAILTO=""

*/1 * * * * root /usr/local/sbin/berofos-workaround
```

1.13.4 Upgrading from XiVO 1.2.3

1. There is an issue with `xivo-libsccp` and `pf-xivo-base-config` during an upgrade from 1.2.3:

```
dpkg: error processing /var/cache/apt/archives/pf-xivo-base-config_13%3a1.2.4-1_all.deb (--ur
trying to overwrite '/etc/asterisk/sccp.conf', which is also in package xivo-libsccp 1.2.3.1-
...
Errors were encountered while processing:
/var/cache/apt/archives/pf-xivo-base-config_13%3a1.2.4-1_all.deb
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

2. You have to remove `/var/lib/dpkg/info/xivo-libsccp.conffiles`:

```
rm /var/lib/dpkg/info/xivo-libsccp.conffiles
```

3. You have to edit `/var/lib/dpkg/info/xivo-libsccp.list` and remove the following line:

```
/etc/asterisk/sccp.conf
```

4. and remove `/etc/asterisk/sccp.conf`:

```
rm /etc/asterisk/sccp.conf
```

5. Now, you can launch `xivo-upgrade` to finish the upgrade process

1.13.5 CTI server is frozen and won't come back online

You must ensure that the partition containing `/var` always has at least 100 MiB of free disk space. If it does not, the symptoms are:

- the CTI server is frozen after logging/unlogging an agent or adding/removing a member from a queue.
- trying to log/unlog an agent via a phone is not possible

To get the system back on tracks after freeing some space in `/var`, you must do:

```
xivo-service restart
```

1.13.6 Agents receiving two ACD calls

An agent can sometimes receive more than 1 ACD call at the same time, even if the queues he's in have the "ringinuse" parameter set to no (default).

This behaviour is caused by a bug in asterisk: https://issues.asterisk.org/jira/browse/ASTERISK-16115

It's possible to workaround this bug in XiVO by adding an agent *subroutine*. The subroutine can be either set globally or per agent:

```
[pre-limit-agentcallback]
exten = s,1,NoOp()
same  =   n,Set(LOCKED=${LOCK(agentcallback)})
same  =   n,GotoIf(${LOCKED}?:not-locked,1)
same  =   n,Set(GROUP(agentcallback)=${XIVO_AGENT_ID})
```

```
same  =   n,Set(COUNT=${GROUP_COUNT(${XIVO_AGENT_ID}@agentcallback)})
same  =   n,NoOp(${UNLOCK(agentcallback)})
same  =   n,GotoIf($[ ${COUNT} <= 1 ]?:too-many-calls,1)
same  =   n,Return()

exten = not-locked,1,NoOp()
same  =   n,Log(ERROR,Could not obtain lock)
same  =   n,Wait(0.5)
same  =   n,Hangup()

exten = too-many-calls,1,NoOp()
same  =   n,Log(WARNING,Not calling agent ID/${XIVO_AGENT_ID} because already in use)
same  =   n,Wait(0.5)
same  =   n,Hangup()
```

This workaround only applies to queues with agent members; it won't work for queues with user members.

Also, the subroutine prevent asterisk from calling an agent twice by hanguping the second call. In the agent statistics, this will be shown as a non-answered call by the agent.

1.14 Documentation changelog

CHAPTER 2

Changelog

The *Documentation changelog* is available.

CHAPTER 3

Indices and tables

- *genindex*
- *search*